SDTM-ETL 4.0: Summary of New Features

Author: Jozef Aerts, XML4Pharma

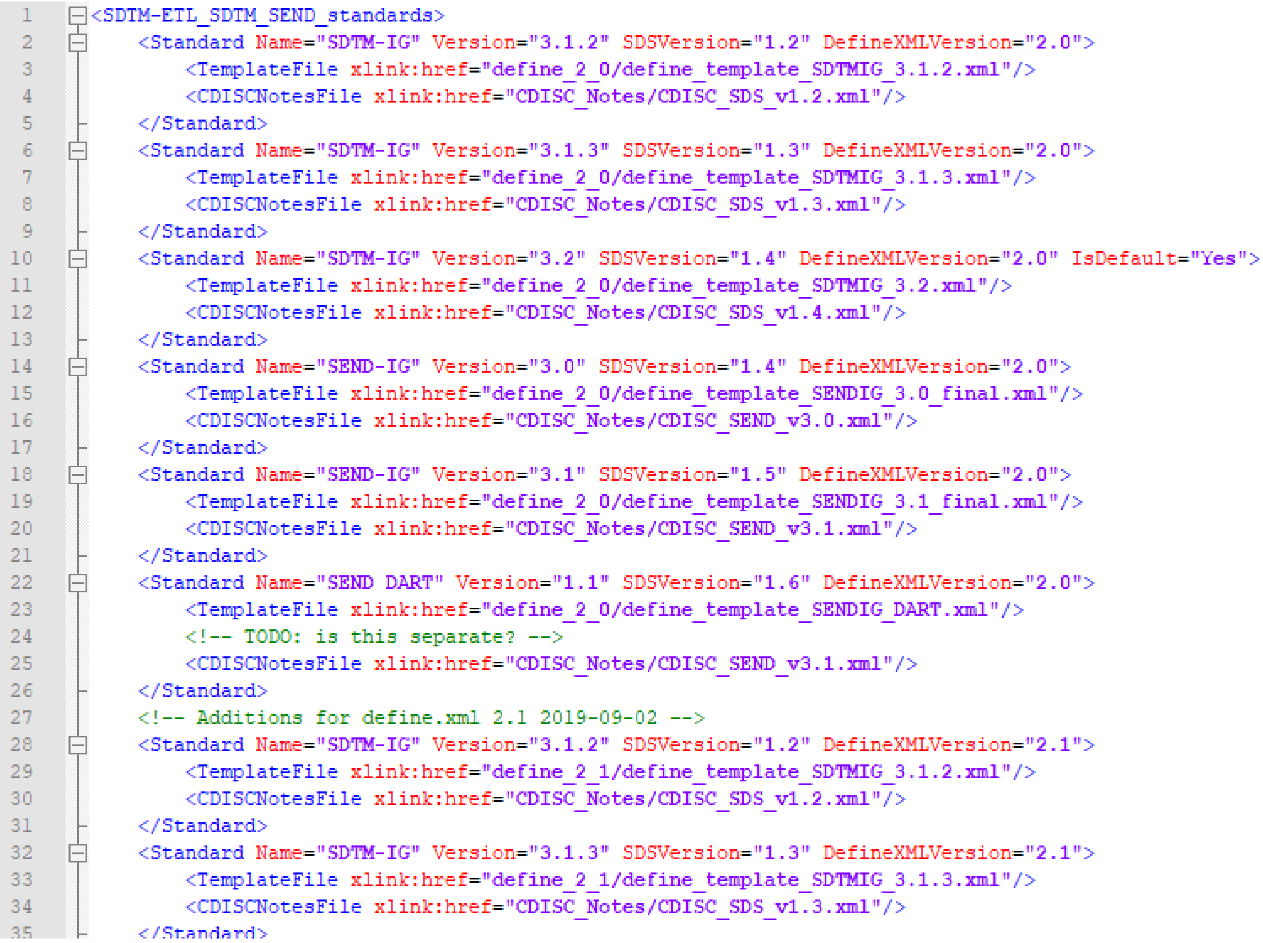
Last update: **2021-12-30**

Summary

This document contains a summary of the most important new features of SDTM-ETL 4.0. There are many minor improvements and new features that are not described in this document, but that can be found in other manuals / tutorials of SDTM-ETL 4.0.

Automated installation of new or additional SDTM/SEND templates

When new versions of the SDTM-IG or SEND-IG are published, an update of the software is not necessary anymore. The initial screen allowing to select which standard (SDTM or SEND) is used and which IG version is now steered by an external file "**SDTM_SEND_standards.xml**". The contents look like:



```
<SDTM-ETL_SDTM_SEND_standards>
    <Standard Name="SDTM-IG" Version="3.1.2" SDSVersion="1.2" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.1.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.2.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.1.3" SDSVersion="1.3" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.1.3.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.3.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.2" SDSVersion="1.4" DefineXMLVersion="2.0" IsDefault="Yes">
        <TemplateFile xlink:href="define_2_0/define_template_SDTMIG_3.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.4.xml"/>
    </Standard>
    <Standard Name="SEND-IG" Version="3.0" SDSVersion="1.4" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_3.0_final.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.0.xml"/>
    </Standard>
    <Standard Name="SEND-IG" Version="3.1" SDSVersion="1.5" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_3.1_final.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.1.xml"/>
    </Standard>
    <Standard Name="SEND DART" Version="1.1" SDSVersion="1.6" DefineXMLVersion="2.0">
        <TemplateFile xlink:href="define_2_0/define_template_SENDIG_DART.xml"/>
        <!-- TODO: is this separate? -->
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SEND_v3.1.xml"/>
    </Standard>
    <!-- Additions for define.xml 2.1 2019-09-02 -->
    <Standard Name="SDTM-IG" Version="3.1.2" SDSVersion="1.2" DefineXMLVersion="2.1">
        <TemplateFile xlink:href="define_2_1/define_template_SDTMIG_3.1.2.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.2.xml"/>
    </Standard>
    <Standard Name="SDTM-IG" Version="3.1.3" SDSVersion="1.3" DefineXMLVersion="2.1">
        <TemplateFile xlink:href="define_2_1/define_template_SDTMIG_3.1.3.xml"/>
        <CDISCNotesFile xlink:href="CDISC_Notes/CDISC_SDS_v1.3.xml"/>
    </Standard>
```

Each entry "Standard" in this file points to a template for that standard for a specific version of the define.xml (2.0 or 2.1 – see further), and (when available) a file with the "CDISC notes". This means that when a new version of e.g. the SDTM-IG is published, no software update is needed, but only the new template files need to be installed, and a link to it to be added to the file "SDTM_SEND_standards.xml" file. These files will then be provided by XML4Pharma at a low cost.

For example, we added "SEND DART 1.1" templates in the current release in this way. Of course, even when choosing "plain" SEND (3.0 or 3.1), one can always add the "SEND DART" afterwards and merge it with the originally loaded template.
Immediately after it was published, we implemented the SDTM-IG v.3.3 and its corresponding SDTM model v.1.7. When starting up, SDTM-IG v.3.2 however remains the default, as the regulatory authorities such as the FDA do not accept SDTM-IG v.3.3 yet (status May 2020).
Also, as of December 2021, the new **SDTM-IG version 3.4** has become available, and was installed in this way. Also the corresponding PDF documents for SDTM v,2.0 and the SDTM-IG itself were added.
Existing users of SDTM-ETL 4.0 can obtain a free upgrade containing SDTM-IG versions 3.3 and 3.4.

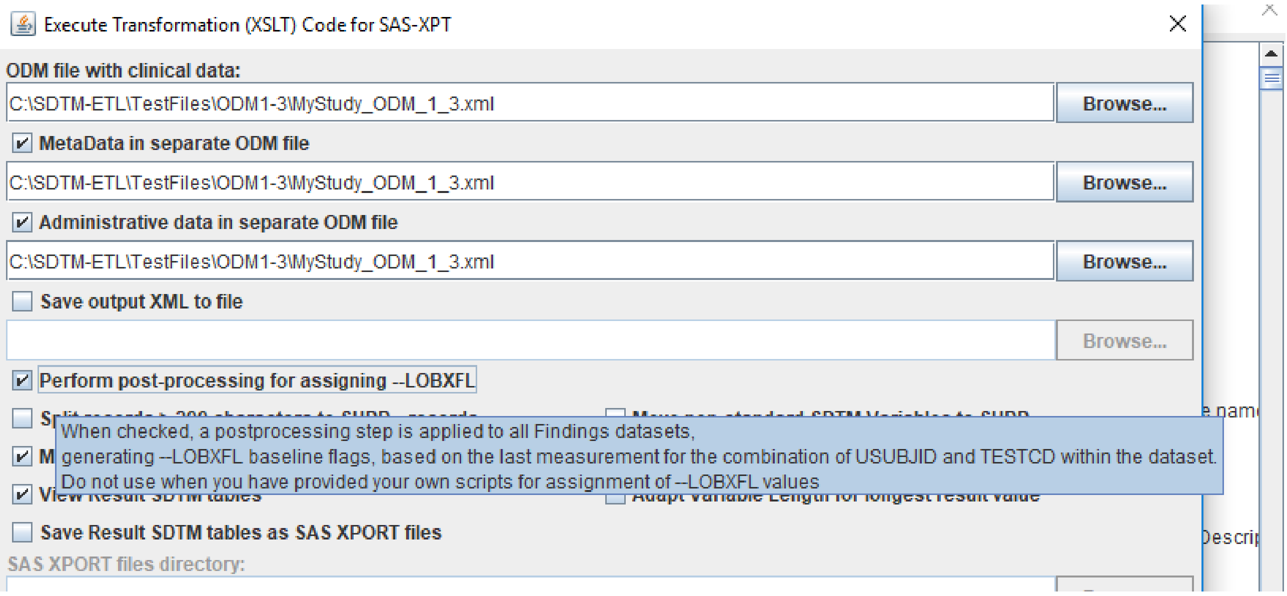
Automated generation of –LOBXFL variable values for SDTM-IG v.3.3 / 3.4

SDTM v.1.7 / SDTM-IG v.3.3 introduces a new set of variables, all ending with "-LOBXFL". These "flag" variables represent the "Last Observation Before Exposure" and are used in almost all Findings domains.
We regret that these variables have been added, as these are essentially "analysis" variables (like other baseline flags), and "analysis variables" essentially should not be present in SDTM (they should go into ADaM). These variables seem to have been added on request of FDA reviewers, who are themselves not able to perform the necessary derivations to get "last observation before treatment" data points. These derivations should however be very easy to perform.

But things are as they are …

As the derivation of values for –LOBXFL is based on that the first exposure is known, the derivation may need an additional run over all generated data. Such an "extra run" has now been implemented in the software.

In the stage where the mappings are executed (both for SAS-XPT as for Dataset-XML format), an extra checkbox is presented:



Execute Transformation (XSLT) Code for SAS-XPT

ODM file with clinical data:
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]

☑ MetaData in separate ODM file
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]

☑ Administrative data in separate ODM file
C:\SDTM-ETL\TestFiles\ODM1-3\MyStudy_ODM_1_3.xml [Browse...]

☐ Save output XML to file
[Browse...]

☑ Perform post-processing for assigning --LOBXFL

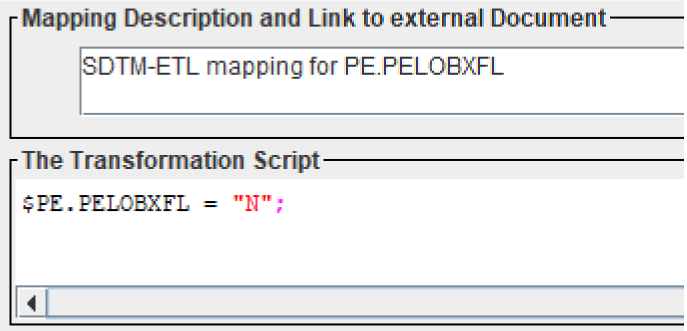
☐ Split records > 200 characters to SUPP-- records ☐ Move non-standard SDTM Variables to SUPP
☑ M
☑ View Result SDTM tables ☐ Adapt Variable Length for longest result value

When checked, a postprocessing step is applied to all Findings datasets,
generating --LOBXFL baseline flags, based on the last measurement for the combination of USUBJID and TESTCD within the dataset.
Do not use when you have provided your own scripts for assignment of --LOBXFL values

☐ Save Result SDTM tables as SAS XPORT files
SAS XPORT files directory:

The checkbox "**Perform post-processing for assigning –LOBXFL**" will indeed initiate a post-processing step, in which the data is analyzed and the very last observation before the first treatment, based on RFXSTDTC ("Date/Time of First Study Treatment") in DM ("Demographics") is assigned for each unique test. When RFXSTDTC is a date only (without part), and the observation has a "datetime of collection" that is on the same date of the first exposure, then it is assumed that the observation is before the treatment. See further how to change this behavior.

Please do remark, that even when this checkbox is checked, you will need to provide a "default" mapping for each –LOBXFL variable, as otherwise the system assumes that the variable is not to be populated and will even not appear in the output files.
So, you will probably want to set the default value to "N" (meaning "no")



Mapping Description and Link to external Document

SDTM-ETL mapping for PE.PELOBXFL

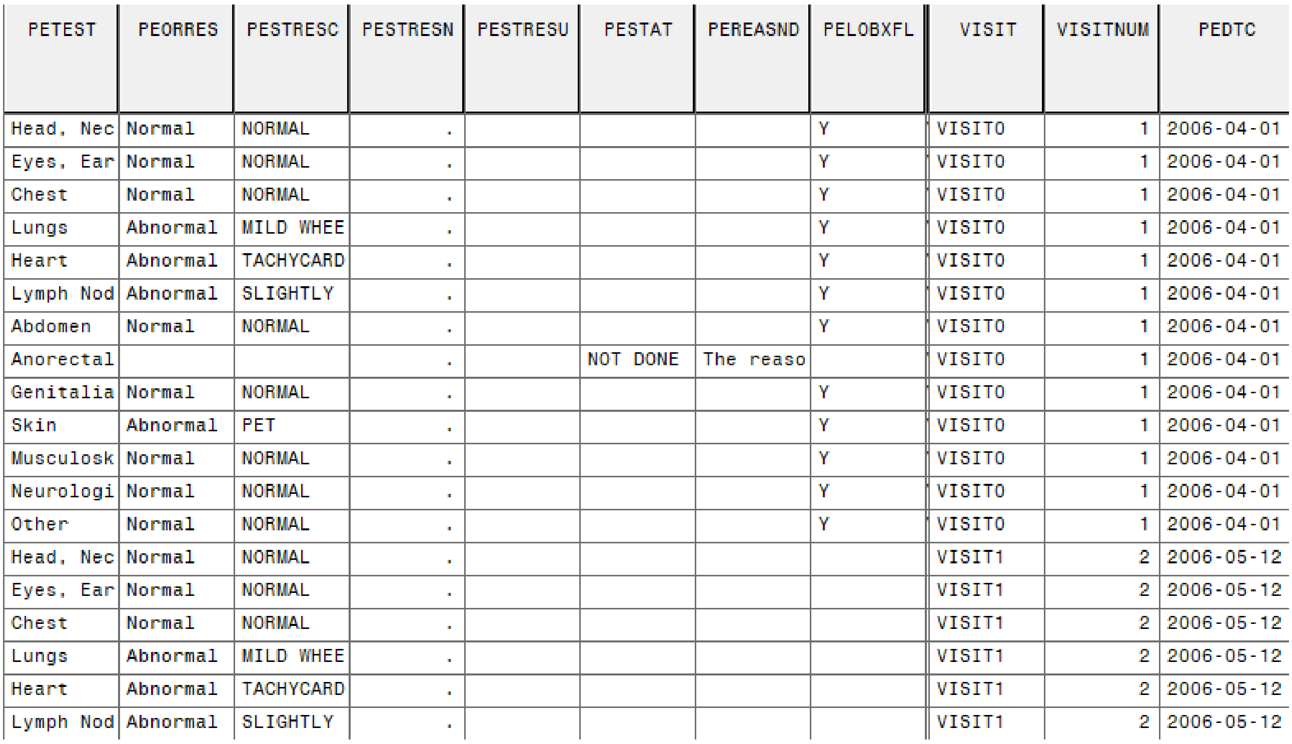
The Transformation Script

```
$PE.PELOBXFL = "N";
```

stating that the initial value (before the post-processing step) is the "no" value.
Remark that in case of output to Dataset-XML, it is not allowed to set the "default" value to the empty value (e.g. $VSLOBXFL = '') as this will not create LOBXFL data points in the intermediate file, as in Dataset-XML, empty values are just not "printed out". In case you only want to generate XPT files, the use of setting the default value to the empty value is unproblematic.

Also note that the SDTM-IG explicitly allows to have an "N" value for -LOBXFL variables.

An example outcome of using the "post-processing", here for the PE dataset, is:



PETEST	PEORRES	PESTRESC	PESTRESN	PESTRESU	PESTAT	PEREASND	PELOBXFL	VISIT	VISITNUM	PEDTC
Head, Nec	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Eyes, Ear	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Chest	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Lungs	Abnormal	MILD WHEE	.				Y	VISIT0	1	2006-04-01
Heart	Abnormal	TACHYCARD	.				Y	VISIT0	1	2006-04-01
Lymph Nod	Abnormal	SLIGHTLY	.				Y	VISIT0	1	2006-04-01
Abdomen	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Anorectal			.		NOT DONE	The reaso		VISIT0	1	2006-04-01
Genitalia	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Skin	Abnormal	PET	.				Y	VISIT0	1	2006-04-01
Musculosk	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Neurologi	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Other	Normal	NORMAL	.				Y	VISIT0	1	2006-04-01
Head, Nec	Normal	NORMAL	.					VISIT1	2	2006-05-12
Eyes, Ear	Normal	NORMAL	.					VISIT1	2	2006-05-12
Chest	Normal	NORMAL	.					VISIT1	2	2006-05-12
Lungs	Abnormal	MILD WHEE	.					VISIT1	2	2006-05-12
Heart	Abnormal	TACHYCARD	.					VISIT1	2	2006-05-12
Lymph Nod	Abnormal	SLIGHTLY	.					VISIT1	2	2006-05-12

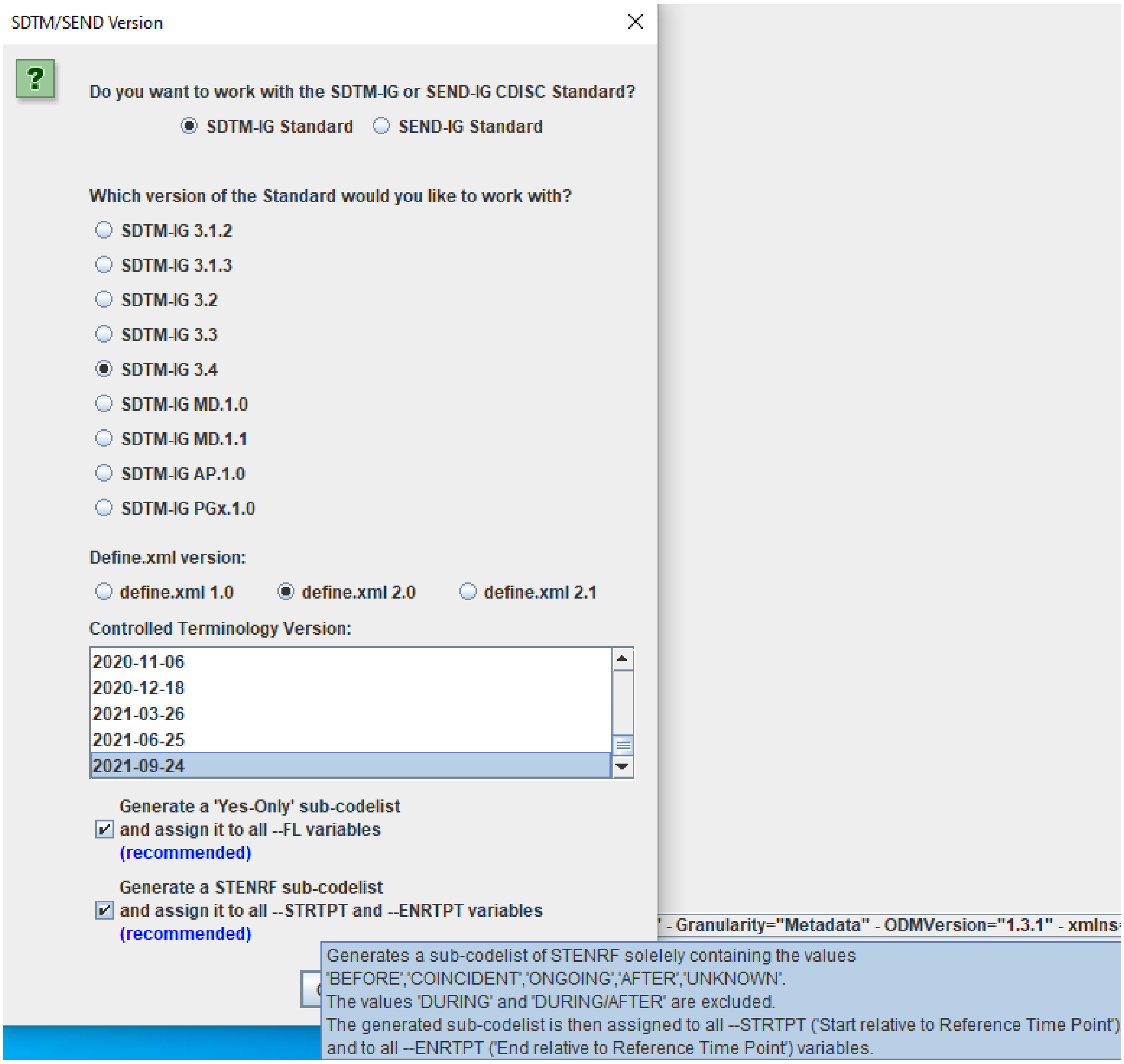
If you do not want this use this post-processing step that uses the above-mentioned assumptions, leave the checkbox "Perform post-processing for assigning –LOBXFL" unchecked, and provide the mapping for each "--LOBXFL" variable yourself.
As you might need the value of "RFXSTDTC", you may declare this variable as a "global variable" (see the manual "Creating and working with Subject Global Variables") and reference it in the mapping. This may e.g. allow you to assign the "last observation" flag to the last value that is at least one day before the first exposure date.

Automated creation of codelist subsets at startup time

One of the problems with CDISC controlled terminology is that the SDTM-IG often constraints them further, just as text, so non-machine-readable. Even though many implementers have asked CDISC to also publish subsets of such codelists, the CT team keeps refusing this.
A famous example is the NY (Yes-No codelist) which contains "N", "Y", "NA", and "U" as allowed values. For "flag variables", variables ending with "FL" however, only the "Y" value is allowed (by constraint). The CDISC-CT team however still refuses to also publish a "Yes only" codelist.
Another example in the "**STENRF**" (Relation to Reference Period) codelist, which has 7 allowed values. This codelist is also assigned by the SDTM-IG to the –STRTPT (Start Relative to Reference Time Point) and -ENRTPT (End Relative to Reference Time Point) variables, but in that case, the values "DURING" and "DURING/AFTER" are forbidden. Also here, the CT team refused to publish a subset.

There are two ways this can be managed. The first is to add such subset codelists to each of the set of codelists quarterly by CDISC. This is the strategy followed by Pinnacle21 (but it often takes them several months to do so, at least for the "Community" version). It is however questionable whether editing by CDISC published codelists is morally acceptable. We don't think so.
The second possibility is to generate such a "subset-codelist" "on the fly" when loading the CDISC-CT as published by CDISC, which does not rely on editing existing codelists.
This is the strategy we have chosen.

When the choice of the SDTM/SEND version is due, the dialog now shows an extra panel with two checkboxes, the first asking whether a "Yes only" codelist should be generated, and automatically assign this codelist to all "flag" variables, all ending with "FL", like all the "baseline flags".
The second checkbox is about generating a subset of the "STENRF" codelist to the 5 values(without "DURING" and "DURING/AFTER", and automatically assign this subset-codelist to all "-STRTPT" and "-ENRTPT" variables:



SDTM/SEND Version

Do you want to work with the SDTM-IG or SEND-IG CDISC Standard?

● SDTM-IG Standard ○ SEND-IG Standard

Which version of the Standard would you like to work with?

○ SDTM-IG 3.1.2
○ SDTM-IG 3.1.3
○ SDTM-IG 3.2
○ SDTM-IG 3.3
● SDTM-IG 3.4
○ SDTM-IG MD.1.0
○ SDTM-IG MD.1.1
○ SDTM-IG AP.1.0
○ SDTM-IG PGx.1.0

Define.xml version:

○ define.xml 1.0 ● define.xml 2.0 ○ define.xml 2.1

Controlled Terminology Version:

2020-11-06
2020-12-18
2021-03-26
2021-06-25
2021-09-24

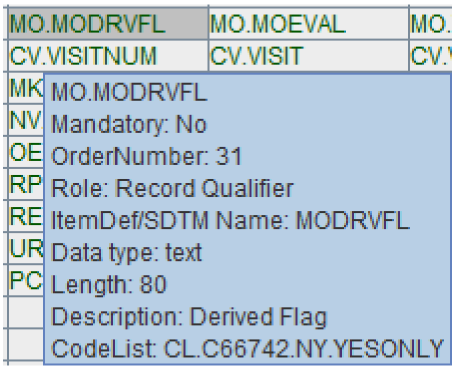
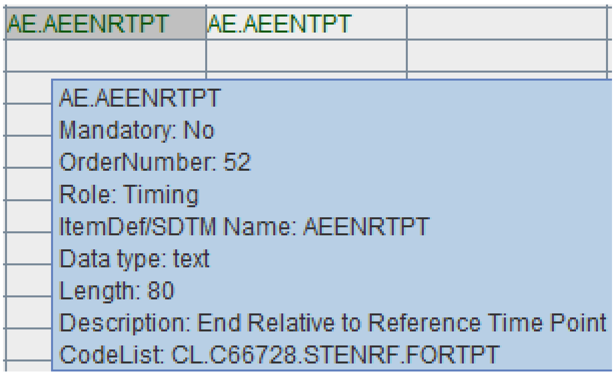
☑ Generate a 'Yes-Only' sub-codelist and assign it to all --FL variables (recommended)

☑ Generate a STENRF sub-codelist and assign it to all --STRTPT and --ENRTPT variables (recommended)

' - Granularity="Metadata" - ODMVersion="1.3.1" - xmlns

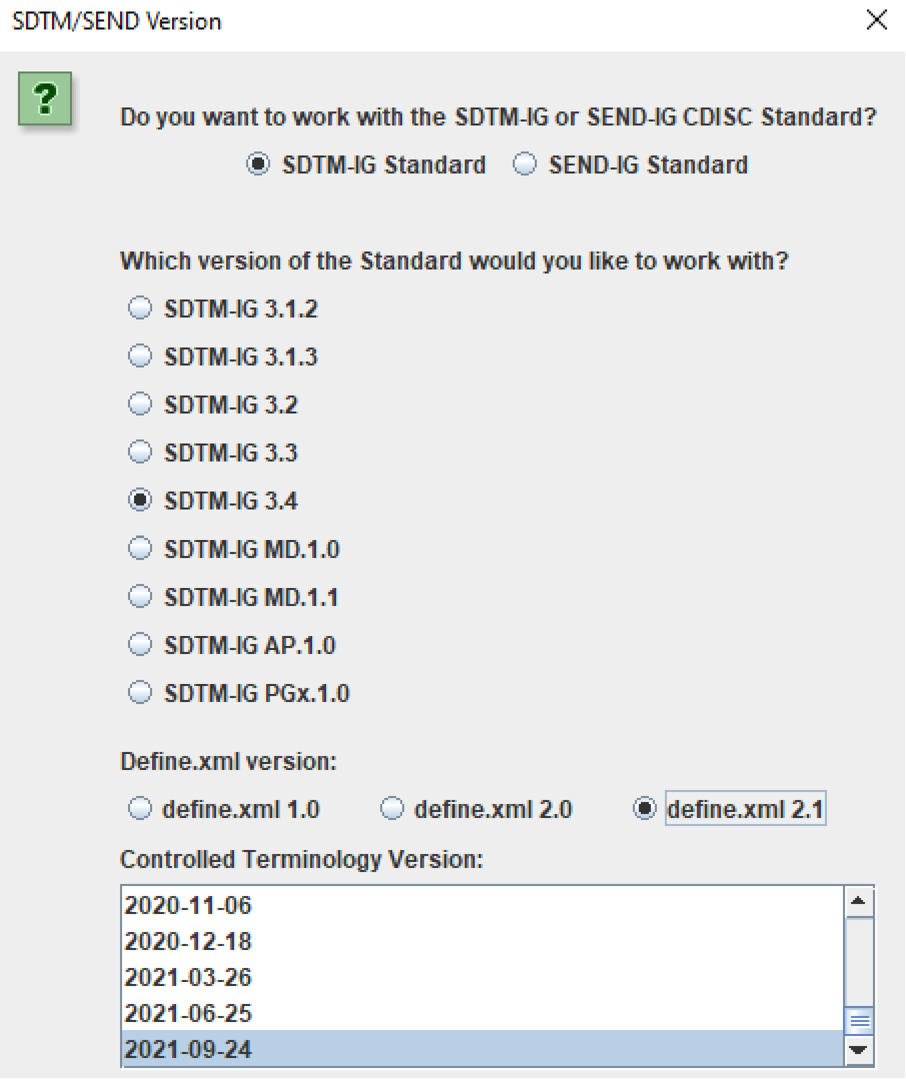
Generates a sub-codelist of STENRF solelely containing the values 'BEFORE','COINCIDENT','ONGOING','AFTER','UNKNOWN'.
The values 'DURING' and 'DURING/AFTER' are excluded.
The generated sub-codelist is then assigned to all --STRTPT ('Start relative to Reference Time Point') and to all --ENRTPT ('End relative to Reference Time Point') variables.

The result when both these checkboxes are checked, is for example, for an -FL variable and for an -ENRTPT variable:

 MO.MODRVFL MO.MOEVAL MO. CV.VISITNUM CV.VISIT CV. MK NV OE RP RE UR PC MO.MODRVFL Mandatory: No OrderNumber: 31 Role: Record Qualifier ItemDef/SDTM Name: MODRVFL Data type: text Length: 80 Description: Derived Flag CodeList: CL.C66742.NY.YESONLY "Yes Only" codelist assigned to MODRVFL	 AE.AEENRTPT Mandatory: No OrderNumber: 52 Role: Timing ItemDef/SDTM Name: AEENRTPT Data type: text Length: 80 Description: End Relative to Reference Time Point CodeList: CL.C66728.STENRF.FORTPT Subset-STENRF ("for TPT") codelist assigned to AEENRTPT

Define.xml 2.1 Implementation

The major feature is that define.xml 2.1 is fully implemented.
As the FDA has just begun starting version 2.1 (status December 2021), version 2.0 remains the default:



SDTM/SEND Version

Do you want to work with the SDTM-IG or SEND-IG CDISC Standard?

● SDTM-IG Standard ○ SEND-IG Standard

Which version of the Standard would you like to work with?

○ SDTM-IG 3.1.2
○ SDTM-IG 3.1.3
○ SDTM-IG 3.2
○ SDTM-IG 3.3
● SDTM-IG 3.4
○ SDTM-IG MD.1.0
○ SDTM-IG MD.1.1
○ SDTM-IG AP.1.0
○ SDTM-IG PGx.1.0

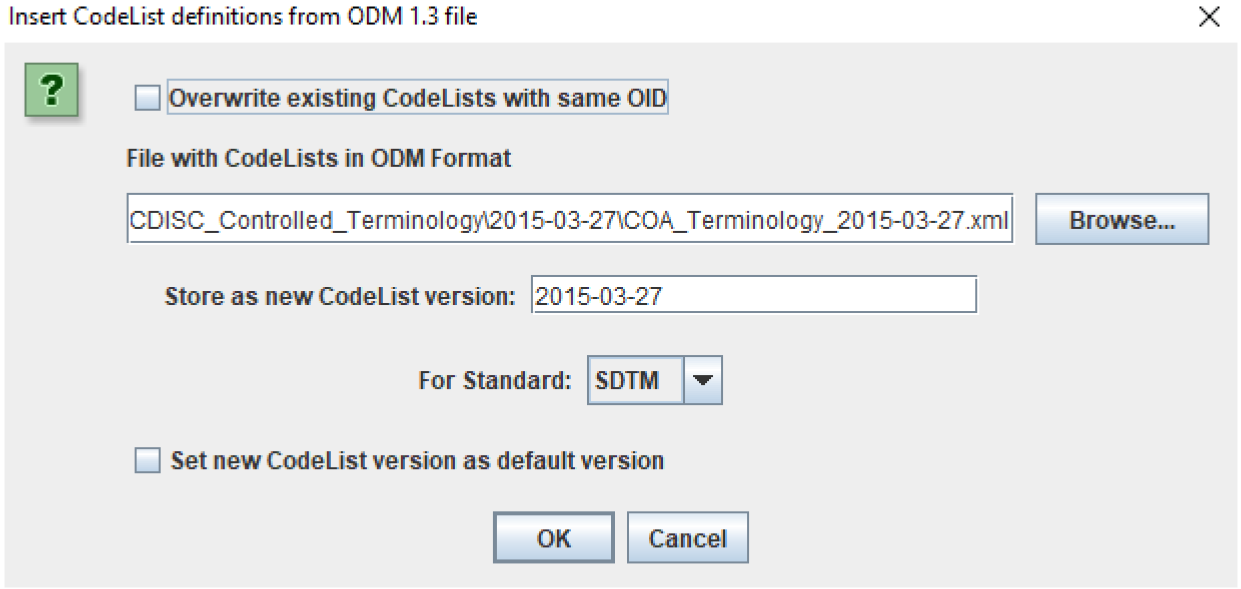
Define.xml version:

○ define.xml 1.0 ○ define.xml 2.0 ● define.xml 2.1

Controlled Terminology Version:

2020-11-06
2020-12-18
2021-03-26
2021-06-25
2021-09-24

One of the new features in the Define-XML standard v.2.1 is that for each domain, it can be defined at the level of the domain/dataset which version of the SDTM-IG (or extension of it, like "associated persons") is used. Also, and that is very useful, for each codelist loaded, it can be defined what the codelist version is.
For example, if additional codelists are loaded from file, using the menu "Insert – CodeList definitions from File", and a codelist (as ODM-XML) is selected, the following dialog appears (example):



Insert CodeList definitions from ODM 1.3 file

□ Overwrite existing CodeLists with same OID

File with CodeLists in ODM Format

CDISC_Controlled_Terminology\2015-03-27\COA_Terminology_2015-03-27.xml

Store as new CodeList version: 2015-03-27

For Standard: SDTM

□ Set new CodeList version as default version

The newly loaded set of codelists can then be set as the "default version", or the already loaded codelists can remain the "default".

Further details about working with the Define-XML v.2.1 standard is provided in a separate tutorial.

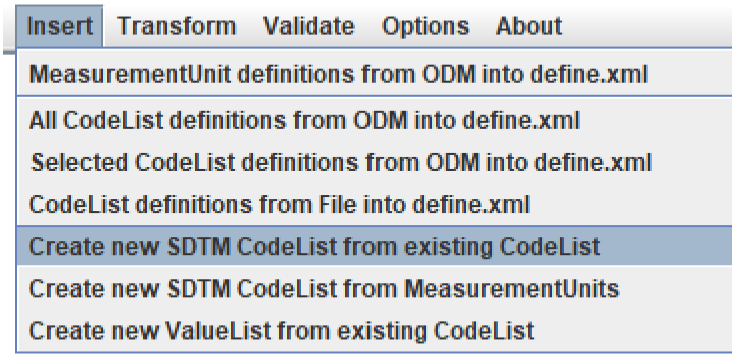
Full support for Chinese and Japanese characters using UTF-8 encoding

The Chinese regulatory authorities NMPA have recently published new guidelines, requiring that labels for datasets, and variable values for a number of domains being submitted in the Chinese language preferably using UTF-8 encoding. As the underlying technology of SDTM-ETL is based on XML, non-European languages using UTF-8 were already for a long time supported, especially when generating the datasets in modern CDISC Dataset-XML format. As of SDTM-ETL 4.0, this support has been extended to SAS Transport 5 format[1].
This is all hidden to the user: the system itself knowns when to encode characters as UTF-8, and when to encode as ASCII. The only thing users need to take care of is that labels with Chinese characters do not exceed 13 characters (corresponding to 40 bytes), and variable values with Chinese characters do not exceed 66 characters (corresponding to 200 bytes).

[1] It is however now already clear that the requirement for SAS XPT with Chinese characters is the worst possible choice, and will lead to huge problems at NMPA. This is however recognized by NMPA yet. Therefore, we are discussing the possibility with NMPA of allowing the sponsor the choice between modern Dataset-XML format or old SAS-XPT format for electronic submissions.

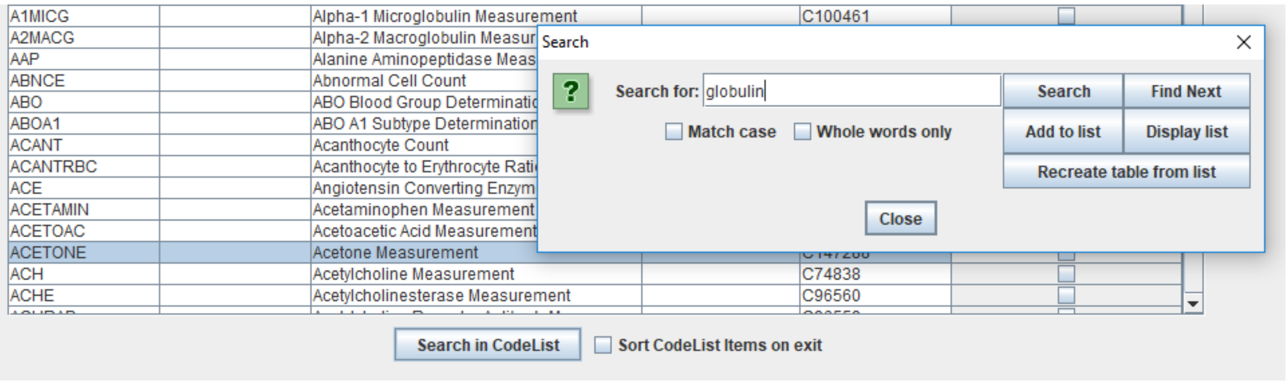
Improved "New codelist from existing CodeList" dialog and functionality

The dialog that appears when using the menu "Insert – New CodeList from existing CodeList" has obtained a lot more functionality that makes it even more easy to generate subset-codelists and to extend codelists:


Insert
Transform
Validate
Options
About
MeasurementUnit definitions from ODM into define.xml
All CodeList definitions from ODM into define.xml
Selected CodeList definitions from ODM into define.xml
CodeList definitions from File into define.xml
Create new SDTM CodeList from existing CodeList
Create new SDTM CodeList from MeasurementUnits
Create new ValueList from existing CodeList

It was already possible to use the "Search" button to find a specific codelist, and then "click through" (i.e. a "next" functionality) until the desired codelist to be subsetted is found, but one can now also use the new "Search in CodeList" button, to find entries in the existing codelist.

For example, when the "Search in CodeList" button is clicked and "globulin" is set as the search term in the dialog that pops up:



A1MICG		Alpha-1 Microglobulin Measurement		C100461	□
A2MACG		Alpha-2 Macroglobulin Measur			
AAP		Alanine Aminopeptidase Meas			
ABNCE		Abnormal Cell Count			
ABO		ABO Blood Group Determinatic			
ABOA1		ABO A1 Subtype Determination			
ACANT		Acanthocyte Count			
ACANTRBC		Acanthocyte to Erythrocyte Rati			
ACE		Angiotensin Converting Enzym			
ACETAMIN		Acetaminophen Measurement			
ACETOAC		Acetoacetic Acid Measurement			
ACETONE		Acetone Measurement			□
ACH		Acetylcholine Measurement		C74838	□
ACHE		Acetylcholinesterase Measurement		C96560	□

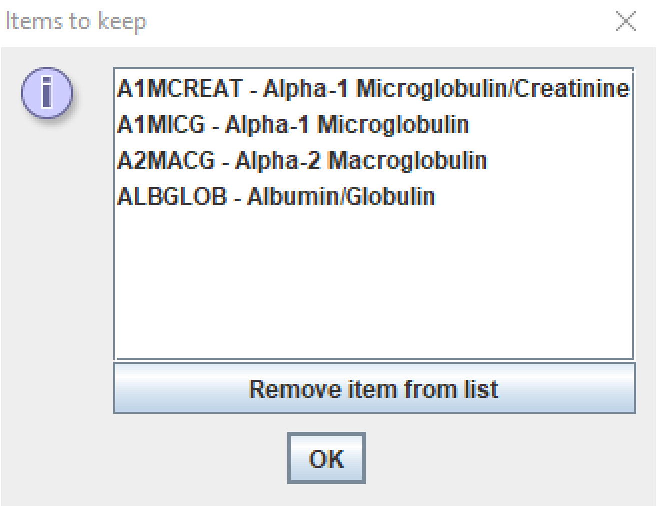
Search

Search for: globulin

□ Match case □ Whole words only

□ Sort CodeList Items on exit

one can then search for all occurrences of "globulin" in that codelist using the "Search" and "Find Next" buttons. Upon each "hit", one can then use the button "Add to list", to add the found entry to the new codelist. With the "Display list" button, one can then display the list of those added, e.g.:

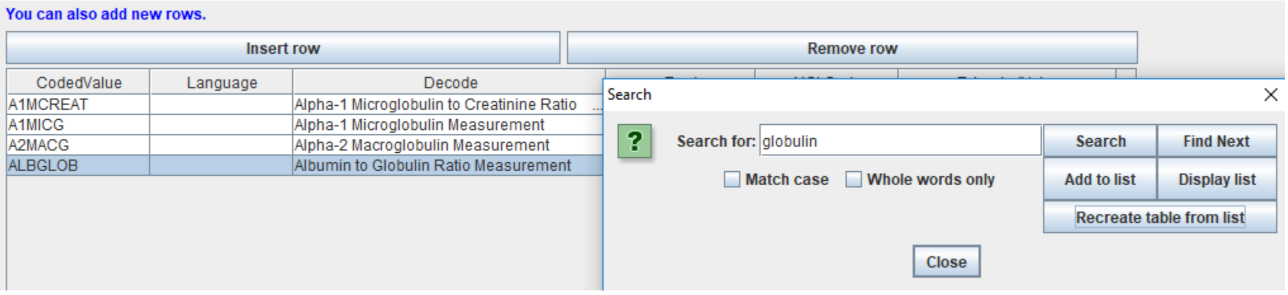


Items to keep

A1MCREAT - Alpha-1 Microglobulin/Creatinine
A1MICG - Alpha-1 Microglobulin
A2MACG - Alpha-2 Macroglobulin
ALBGLOB - Albumin/Globulin

In this dialog, one can also remove items from the list (e.g. when one has made a mistake") using the button "Remove Item from list".

When your "positive" list is ready, and after clicking "OK" in the "Display list" dialog, and the button "Recreate table from list" is subsequently clicked, only the ones from the "positive" list are retained and added to the new codelist:


You can also add new rows.

CodedValue	Language	Decode
A1MCREAT		Alpha-1 Microglobulin to Creatinine Ratio
A1MICG		Alpha-1 Microglobulin Measurement
A2MACG		Alpha-2 Macroglobulin Measurement
ALBGLOB		Albumin to Globulin Ratio Measurement

Search

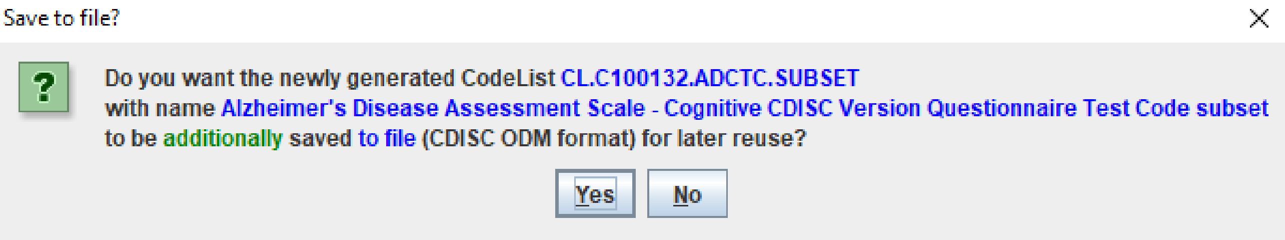
Search for: globulin

☐ Match case ☐ Whole words only

One can then still add or remove items to/from the newly created codelist, e.g. to extend the codelist with non-CDISC terms.
Also remark that the NCI code appears in the table when the starting codelist is a CDISC codelist.

This newly added "wizard" makes it even more easy to quickly generate subsets of codelists with or without sponsor extensions.

Also new is that when the new codelist is then generated (and added to the underlying define.xml), the user is also asked whether he/she wants to store this newly generated codelist to an external file for future reuse (or even for use in study design!).


Save to file?

Do you want the newly generated CodeList CL.C100132.ADCTC.SUBSET
with name Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code subset
to be additionally saved to file (CDISC ODM format) for later reuse?

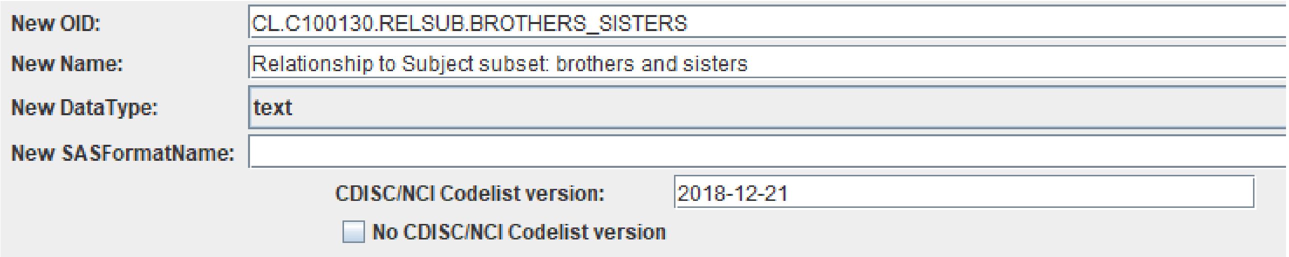
Such a generated file can then be read in again in mappings for other studies using the menu "Insert – CodeList Definitions from File into define.xml".

Also new is that when a new term is added manually (using "Insert Row"), the checkbox "Extended Value" is checked automatically. It can of course be unchecked again when the manually added term is an official CDISC term for that codelist. In that case, also the NCI code must be entered.

Additionally, when a cell in the SDTM/SEND table is selected, and a codelist is already associated with that SDTM/SEND variable (e.g. "LBTESTCD"), then the codelist is pre-selected when using the menu "Insert – Create new CodeList from existing CodeList".

When a codelist is generated from an existing CDISC-CT codelist (which has an NCI code), and none of the original items remains in the edited codelist (i.e. only new values are present), then the user is asked whether the NCI code of the codelist may be dropped (this will usually be the case) or whether it should be retained anyway this although none of the items is a CDISC term anymore.

Finally, when define.xml 2.1 is used, the user is asked to confirm the codelist version of the newly generated codelist. It is also possible to assign another codelist version to the newly generated codelist (do however use with care!) or not to assign a CDISC/NCI version to the subset or new codelist. The latter may be the case when the new codelist is a "sponsor-defined" codelist.



New OID:	CL.C100130.RELSUB.BROTHERS_SISTERS
New Name:	Relationship to Subject subset: brothers and sisters
New DataType:	text
New SASFormatName:	

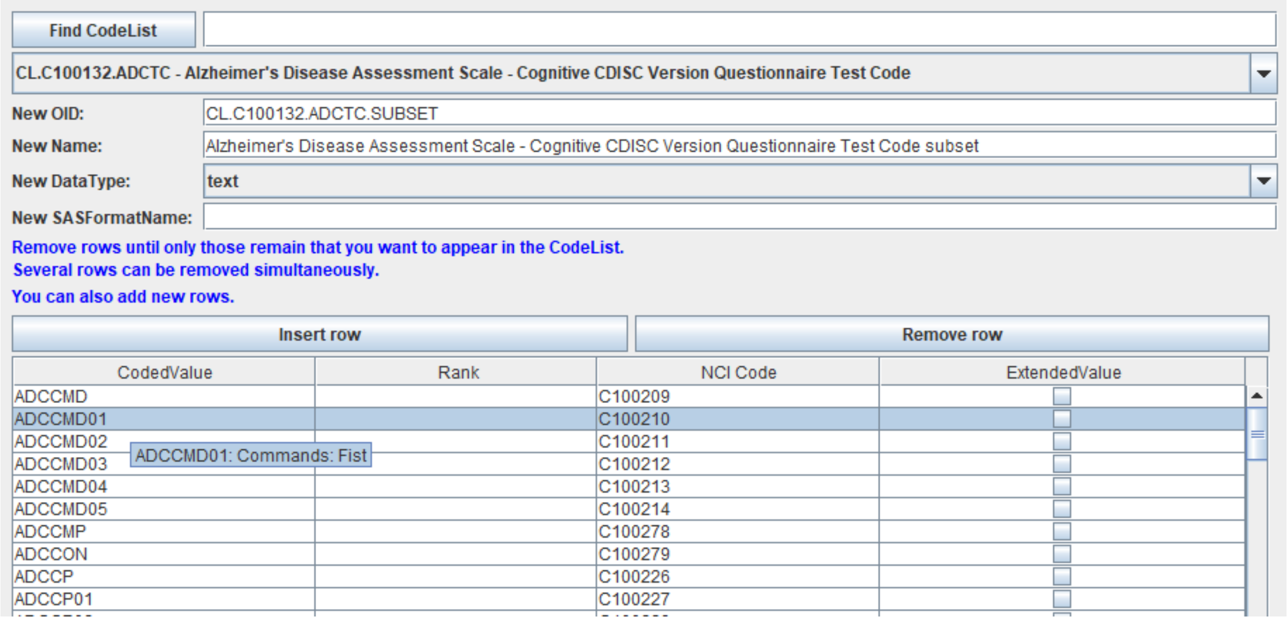
CDISC/NCI Codelist version: 2018-12-21

☐ No CDISC/NCI Codelist version

Subsetting codelists: handling of "Enumerated" codelists

Unfortunately, the CDISC-CT team is more and more publishing its codelists as "enumerated" codelists, even for codelists that contains codes for which "decodes" exist. Typically, these are lists of "test codes" where the "test name" is in a separate list. As sometimes the codes are meaningless, this makes it difficult to create good subset codelists.

For example, for the codelist "*Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code*" (CL.C100132.ADCTC) just contains codes like "ADCCMD" which are hard to interpret when setting up a subset-codelist. The link with the "test name" is only provided through the NCI code. As such, when starting to create a subset, the following dialog and table appears:



CL.C100132.ADCTC - Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code

New OID:	CL.C100132.ADCTC.SUBSET
New Name:	Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code subset
New DataType:	text
New SASFormatName:	

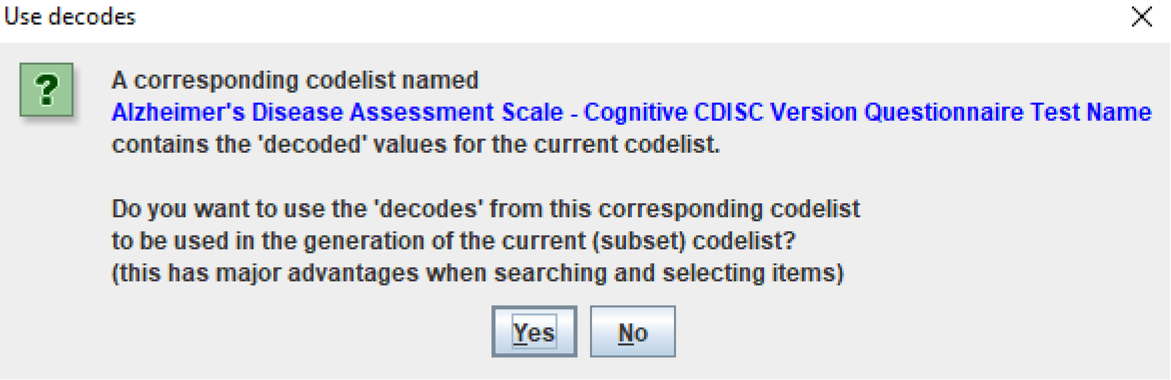
Remove rows until only those remain that you want to appear in the CodeList.
Several rows can be removed simultaneously.
You can also add new rows.

CodedValue	Rank	NCI Code	ExtendedValue
ADCCMD		C100209	☐
ADCCMD01		C100210	☐
ADCCMD02		C100211	☐
ADCCMD03		C100212	☐
ADCCMD04		C100213	☐
ADCCMD05		C100214	☐
ADCCMP		C100278	☐
ADCCON		C100279	☐
ADCCP		C100226	☐
ADCCP01		C100227	☐

ADCCMD01: Commands: Fist

Although the corresponding "test name" appears as a tooltip, using such an "enumerated" codelist is very unpractical when trying to create a subset.

Therefore, when such an "enumerated" codelist is detected, usually with "test codes", for which there also is a corresponding list with "test names", the user is asked whether the codelist may be transformed "on the fly" to a "classic" codelist with codes and decodes (as would be the better practice anyway):

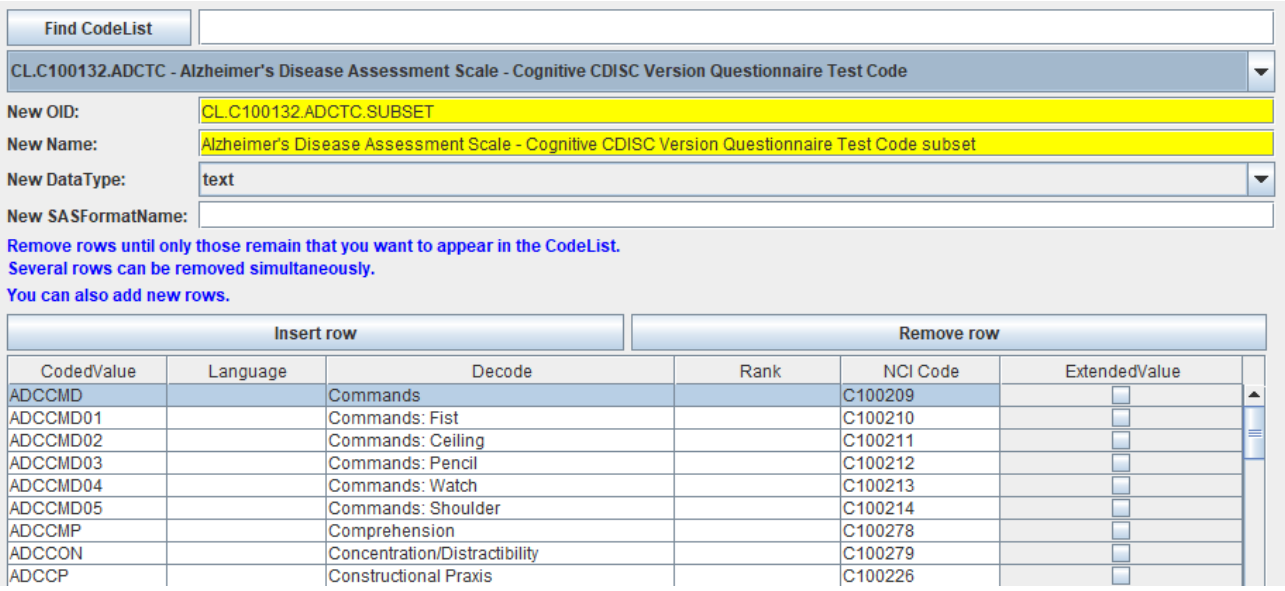


Use decodes

A corresponding codelist named
Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Name
contains the 'decoded' values for the current codelist.

Do you want to use the 'decodes' from this corresponding codelist
to be used in the generation of the current (subset) codelist?
(this has major advantages when searching and selecting items)

When the user then selects "No", the above displayed "enumerated" list appears. When however, "Yes" is selected, the system looks up the corresponding "test names" in the corresponding "enumerated" list with "test names" and adds them to the list with codes, as "decodes".
In the case of very long codelists, these "look ups" may require several seconds to complete.
This then leads to:



CL.C100132.ADCTC - Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code

New OID: CL.C100132.ADCTC.SUBSET

New Name: Alzheimer's Disease Assessment Scale - Cognitive CDISC Version Questionnaire Test Code subset

New DataType: text

New SASFormatName:

Remove rows until only those remain that you want to appear in the CodeList.
Several rows can be removed simultaneously.
You can also add new rows.

CodedValue	Language	Decode	Rank	NCI Code	ExtendedValue
ADCCMD		Commands		C100209	□
ADCCMD01		Commands: Fist		C100210	□
ADCCMD02		Commands: Ceiling		C100211	□
ADCCMD03		Commands: Pencil		C100212	□
ADCCMD04		Commands: Watch		C100213	□
ADCCMD05		Commands: Shoulder		C100214	□
ADCCMP		Comprehension		C100278	□
ADCCON		Concentration/Distractibility		C100279	□
ADCCP		Constructional Praxis		C100226	□

Which is much more practical when generating a subset. When then finishing the subset creation, the subset-codelist is then also stored in the define.xml as a "classic" codelist, and not as a difficult-to-handle "enumerated" codelist. This is also in the advantage of the reviewer at the regulatory authorities, as he/she will see both the codes (test code) and decodes (test name) when inspecting the define.xml.

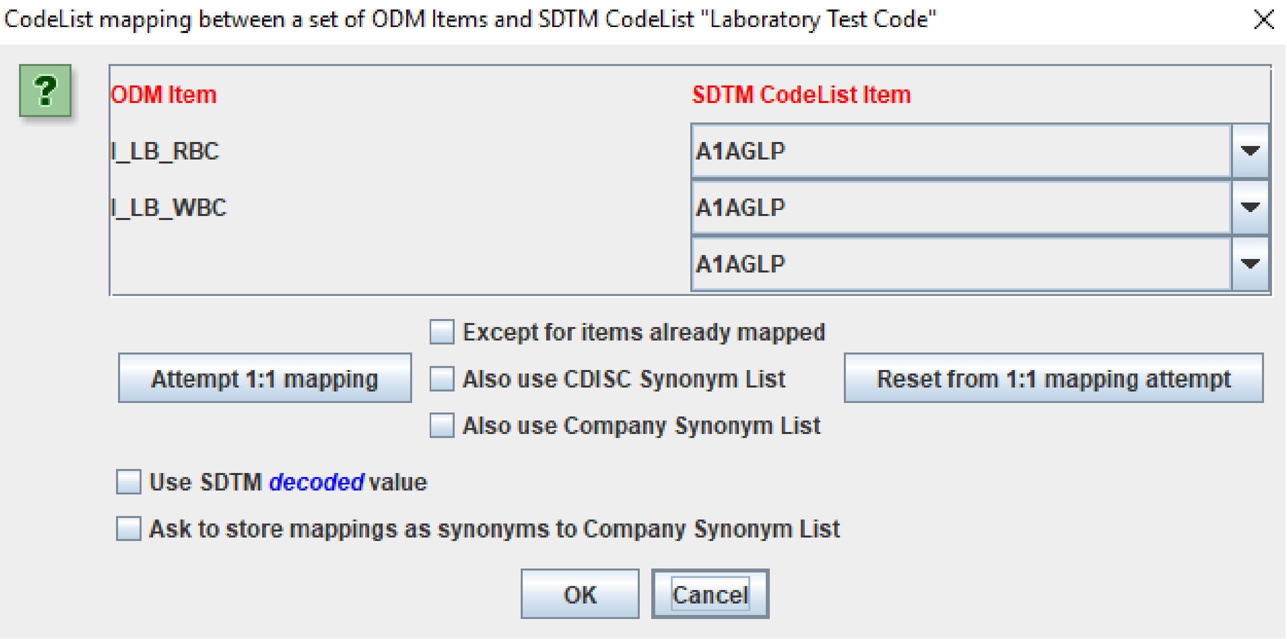
When using the menu "Edit – SDTM/SEND CodeList", the same functionality can also be applied.

For more detailed information, see the document "Subsetting and Extending CodeLists".

Improved CodeList-CodeList mapping

Codelist to codelist mapping can sometimes be tedious because CDISC controlled terminology uses "jargon" that is not always known to the mapper. For example, not anyone may know that "WBC" (the CDISC code) means "White Blood Count" which is also known as "Leukocytes".

When using the codelist to codelist mapper wizard, a few new options are now available for use with the feature "Attempt 1:1 mapping":



CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Code"

ODM Item	SDTM CodeList Item
I_LB_RBC	A1AGLP
I_LB_WBC	A1AGLP
	A1AGLP

☐ Except for items already mapped

☐ Also use CDISC Synonym List

☐ Also use Company Synonym List

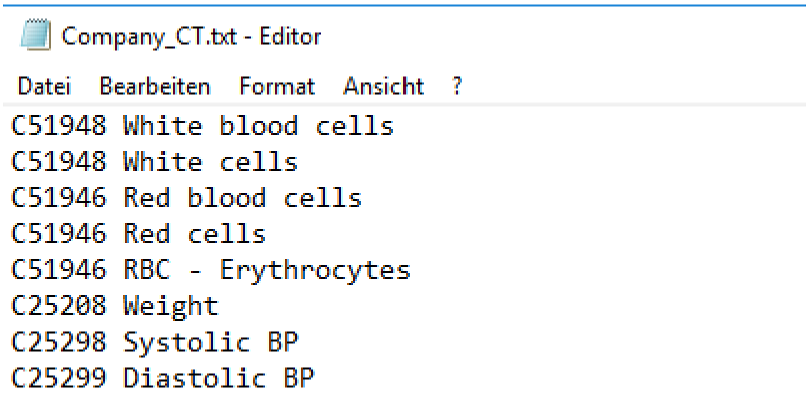
☐ Use SDTM *decoded* value

☐ Ask to store mappings as synonyms to Company Synonym List

One sees two additional checkboxes:

- Also use CDISC Synonym List
- Also use Company Synonym List

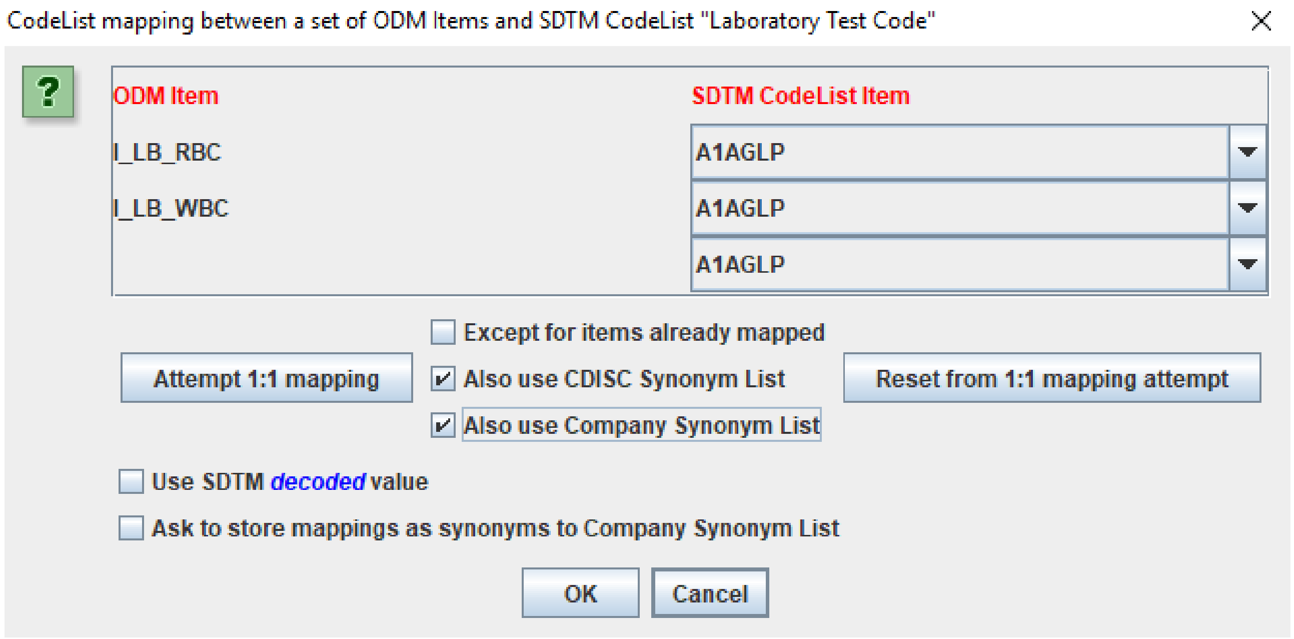
Using the "CDISC Synonym List", when allowing the system to attempt a 1:1 mapping, will also look into the published CDISC synonyms. One can also use a "Company Synonym List" containing the CDISC-NCI codes and locally used synonyms. This list is expected to be located in the folder "Company_CT" and the file name being "Company_CT.txt". Its contents look like:



```
C51948 White blood cells
C51948 White cells
C51946 Red blood cells
C51946 Red cells
C51946 RBC - Erythrocytes
C25208 Weight
C25298 Systolic BP
C25299 Diastolic BP
```

Using the checkbox "Ask to store mappings as synonyms to Company Synonym List" allows to extend this list with new mappings to "locally" used terms.

In our case, using both above mentioned options (checking both checkboxes):



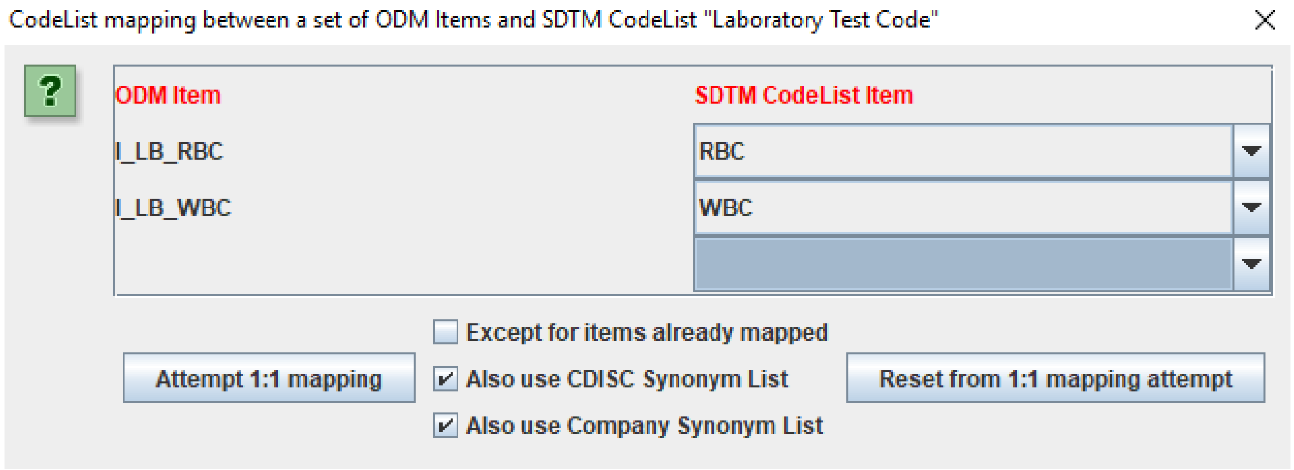
CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Code"

ODM Item	SDTM CodeList Item
I_LB_RBC	A1AGLP
I_LB_WBC	A1AGLP
	A1AGLP

□ Except for items already mapped
☑ Also use CDISC Synonym List
☑ Also use Company Synonym List

□ Use SDTM *decoded* value
□ Ask to store mappings as synonyms to Company Synonym List

and clicking "Attempt 1:1 mapping" leads to:



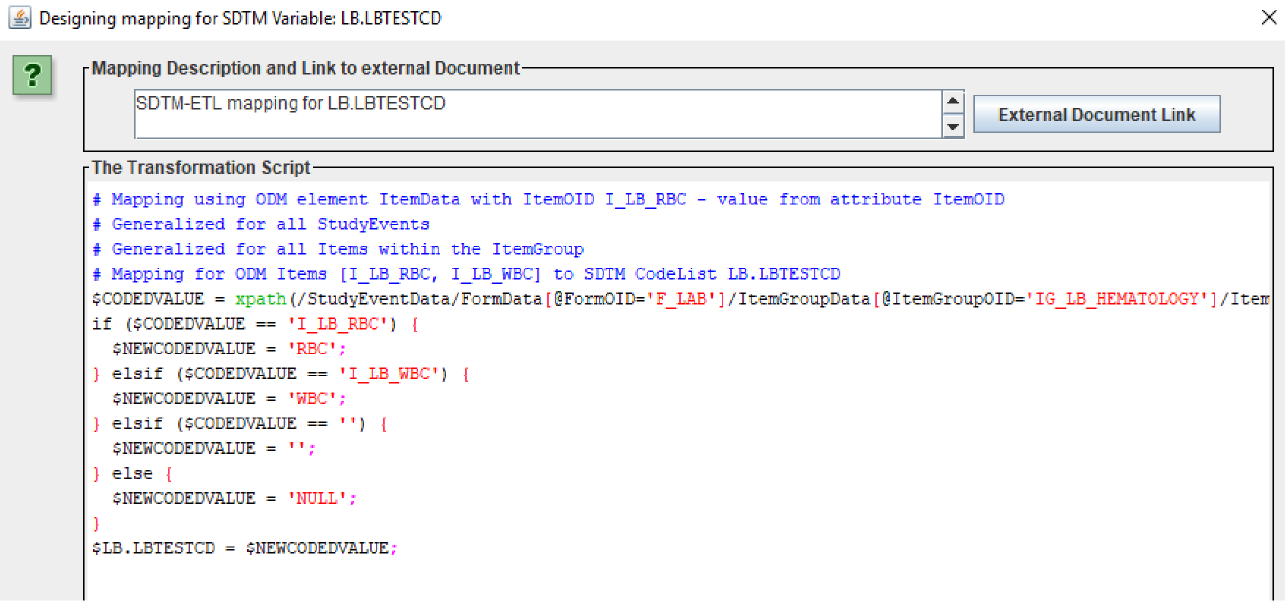
CodeList mapping between a set of ODM Items and SDTM CodeList "Laboratory Test Code"

ODM Item	SDTM CodeList Item
I_LB_RBC	RBC
I_LB_WBC	WBC

□ Except for items already mapped
☑ Also use CDISC Synonym List
☑ Also use Company Synonym List

which in this case is 100% correct.
If one does not like what the system proposes, one can always revert and go to a "manual" mapping, by clicking the "Reset from 1:1 mapping attempt".

The automatically generated script then is:



Designing mapping for SDTM Variable: LB.LBTESTCD

Mapping Description and Link to external Document

SDTM-ETL mapping for LB.LBTESTCD

The Transformation Script

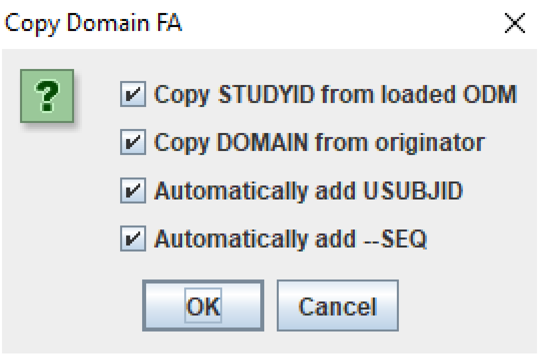
```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/Item
if ($CODEDVALUE == 'I_LB_RBC') {
  $NEWCODEDVALUE = 'RBC';
} elsif ($CODEDVALUE == 'I_LB_WBC') {
  $NEWCODEDVALUE = 'WBC';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$LB.LBTESTCD = $NEWCODEDVALUE;
```

Working with the "Findings About" domain

When SDTM was first developed, there were about 20 domains, and everybody (except a few) was expecting that this number would not increase very much. In the latest SDTM-IG version 3.2 there are over 50 domains, and even that was not sufficient, as it did not cover well the use case of findings that are related to interventions and to events.

For this case, the FA domain "Findings about Events and Interventions" was created.
The SDTM-IG lets implementers the choice between a single FA dataset, and a series of "splitted" datasets for findings about different existing SDTM "Events" and "Interventions" domains. In the latter case one may e.g. have an "FAMH" dataset ("Findings About Medical History") and an "FAAE" ("Findings About Adverse Events") dataset.
The name "splitted" is confusing, as one will usually not split a single FA dataset in datasets such as FAMH and FAAE, but one will create a single "FA" instance for "Medical History" and one for "Adverse Events" right from the start.

As of SDTM-ETL v.4.0, when one "drags-and-drops" the FA row to the bottom (after the last template domain row), or when one selects the FA row from the template and then uses the menu "Edit – Copy Domain" followed by "Edit – Paste Domain", the following dialog is displayed:

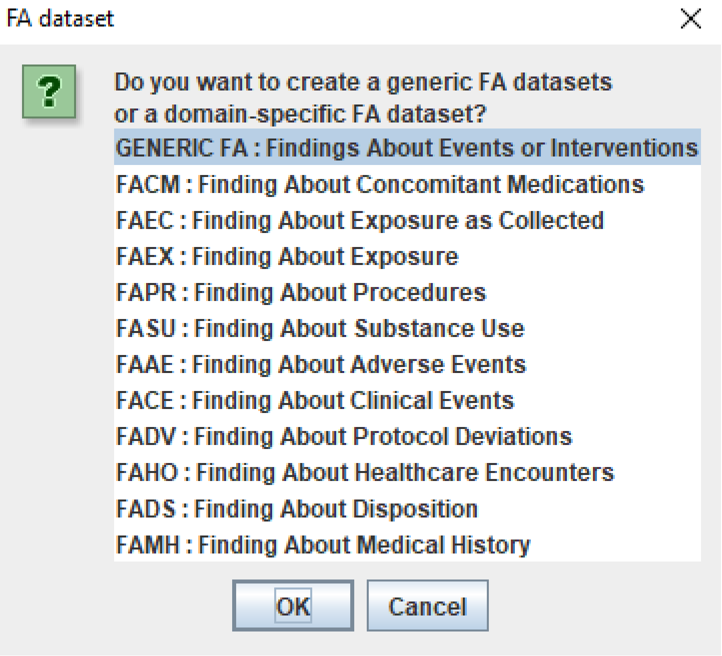


Copy Domain FA

☑ Copy STUDYID from loaded ODM
☑ Copy DOMAIN from originator
☑ Automatically add USUBJID
☑ Automatically add --SEQ

Which is the usual dialog asking whether mappings for STUDYID, DOMAIN, USUBJID, and FASEQ can be automatically generated (recommended when using standard ODM as the source).

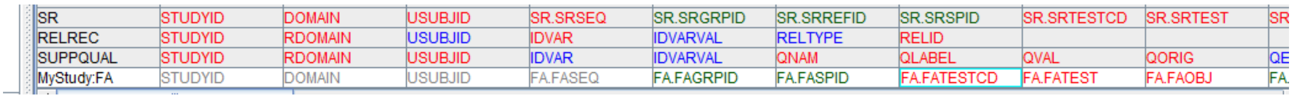
After clicking OK, the following (new) wizard is displayed:


FA dataset

Do you want to create a generic FA datasets
or a domain-specific FA dataset?

GENERIC FA : Findings About Events or Interventions
FACM : Finding About Concomitant Medications
FAEC : Finding About Exposure as Collected
FAEX : Finding About Exposure
FAPR : Finding About Procedures
FASU : Finding About Substance Use
FAAE : Finding About Adverse Events
FACE : Finding About Clinical Events
FADV : Finding About Protocol Deviations
FAHO : Finding About Healthcare Encounters
FADS : Finding About Disposition
FAMH : Finding About Medical History

The software automatically looks up all "Interventions" and "Events" domains in the template, also sponsor-defined ones, and then creates a list of "FA domain specific" domains such as "FACM", FAEC, … The first entry however is "Generic FA", meaning that a single FA instance will be created, which should then contain all "Findings About" entries, independent of what the related domain is. If one selects "Generic FA", a single row is created at the bottom, from which one can start the mappings to a "generic" FA dataset:

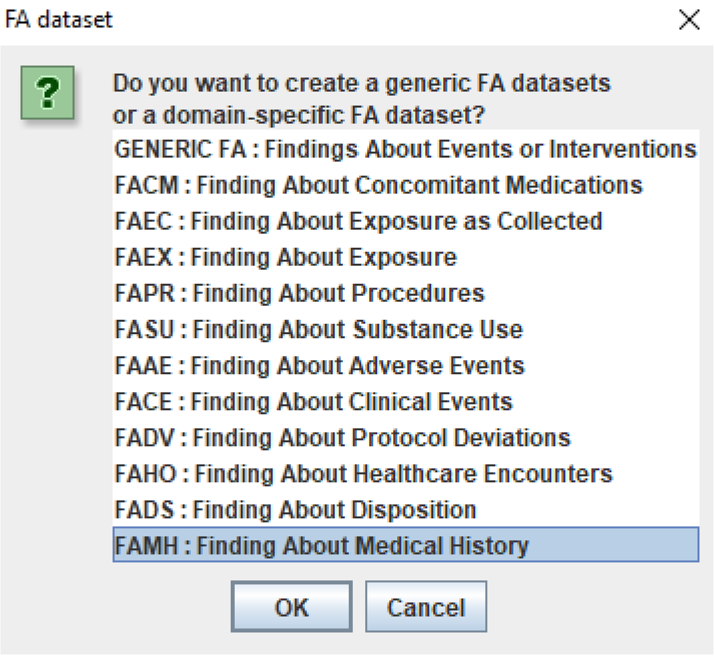


SR	STUDYID	DOMAIN	USUBJID	SR.SRSEQ	SR.SRGRPID	SR.SRREFID	SR.SRSPID	SR.SRTESTCD	SR.SRTEST	SR
RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	RELTYPE	RELID			
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG	QE
MyStudy:FA	STUDYID	DOMAIN	USUBJID	FA.FASEQ	FA.FAGRPID	FA.FASPID	FA.FATESTCD	FA.FATEST	FA.FAOBJ	FA.

This is the simple case and does not deviate from the usual case for any other domain.

One can then start generating the mappings in the usual way.

In a number of cases however, one will want to generate "domain-specific" FA datasets, such as FAMH ("Findings About Medical History") or FACE ("Findings About Clinical Events"). For example, when one wants to create an FAMH dataset instance, one would choose:

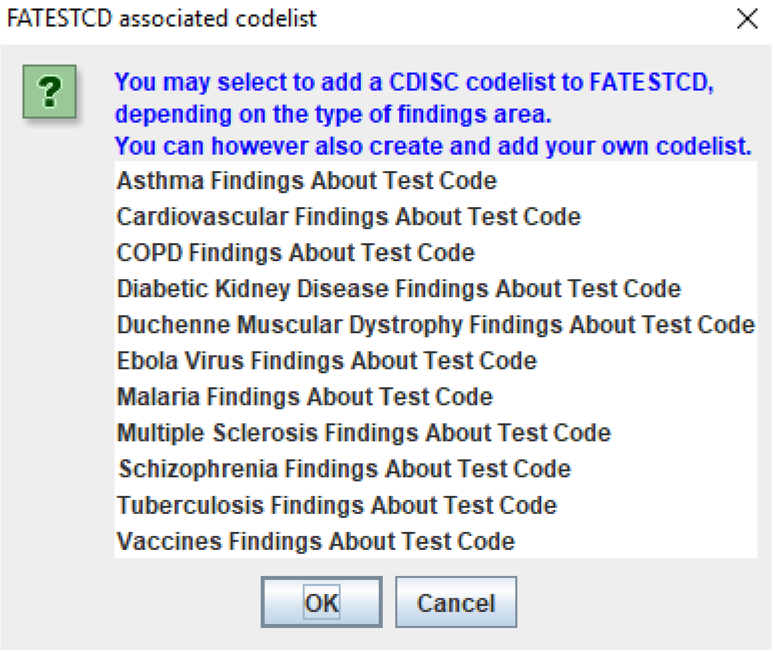


Do you want to create a generic FA datasets
or a domain-specific FA dataset?

- GENERIC FA : Findings About Events or Interventions
- FACM : Finding About Concomitant Medications
- FAEC : Finding About Exposure as Collected
- FAEX : Finding About Exposure
- FAPR : Finding About Procedures
- FASU : Finding About Substance Use
- FAAE : Finding About Adverse Events
- FACE : Finding About Clinical Events
- FADV : Finding About Protocol Deviations
- FAHO : Finding About Healthcare Encounters
- FADS : Finding About Disposition
- FAMH : Finding About Medical History

meaning that an FAMH dataset will be created.

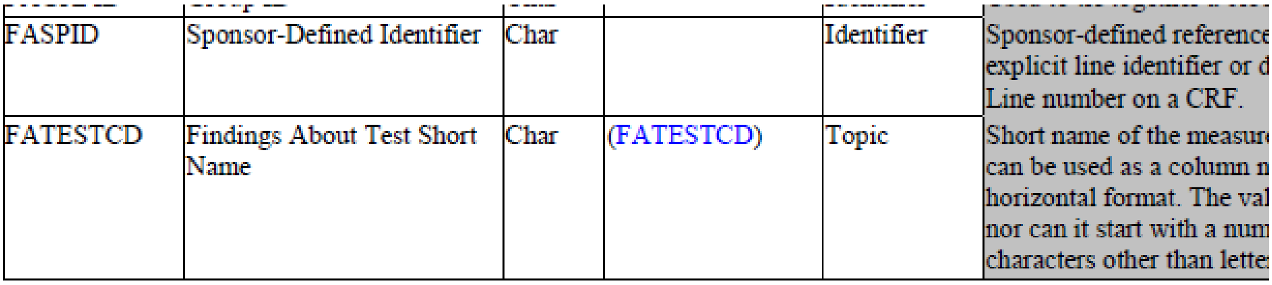
After clicking OK, a new wizard is displayed:



You may select to add a CDISC codelist to FATESTCD,
depending on the type of findings area.
You can however also create and add your own codelist.

- Asthma Findings About Test Code
- Cardiovascular Findings About Test Code
- COPD Findings About Test Code
- Diabetic Kidney Disease Findings About Test Code
- Duchenne Muscular Dystrophy Findings About Test Code
- Ebola Virus Findings About Test Code
- Malaria Findings About Test Code
- Multiple Sclerosis Findings About Test Code
- Schizophrenia Findings About Test Code
- Tuberculosis Findings About Test Code
- Vaccines Findings About Test Code

In the SDTM-IG 3.2, it is stated that the **codelist** "FATESTCD" needs to be used for the variable FATESTCD:

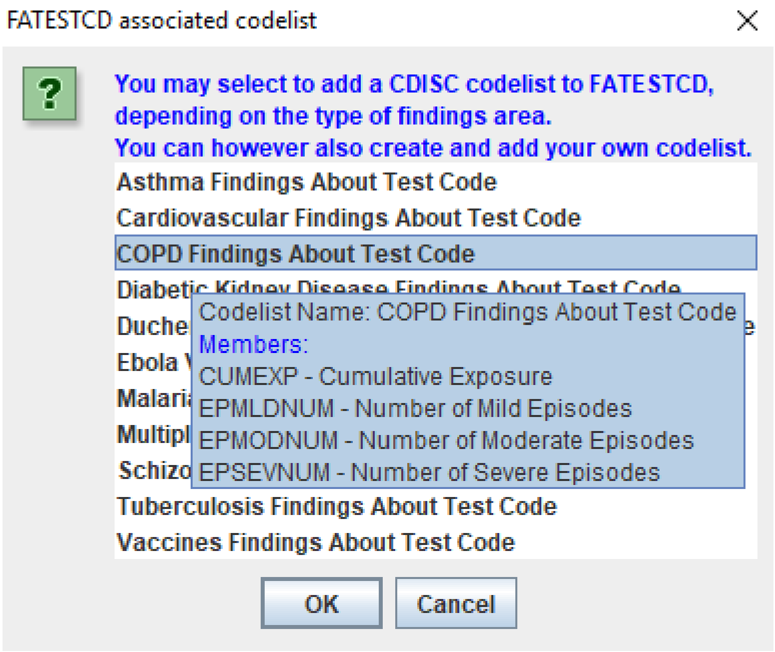


FASPID	Sponsor-Defined Identifier	Char		Identifier	Sponsor-defined reference explicit line identifier or d Line number on a CRF.
FATESTCD	Findings About Test Short Name	Char	(FATESTCD)	Topic	Short name of the measur can be used as a column n horizontal format. The val nor can it start with a num characters other than lette

However, the newer versions of CDISC Controlled Terminology (CDISC-CT) does NOT contain this codelist. Closer inspection of all the CDISC-CT of the last years show that it was deleted in September 2016.

Instead, a number of disease-specific "FATESTCD" codelists has been developed by the CDISC-CT team. All these obey to the pattern "–FATSCD", and the corresponding codelist for "FASTEST" obeys to the pattern "—FATS". For example, for "COPD Findings About Test Code", the identifier is "CPFATSCD" and the NCI code is "C122007".

In the wizard, when hovering the mouse over an entry, all the allowed values for that codelist is displayed. For example, for "CPFATSCD":



FATESTCD associated codelist

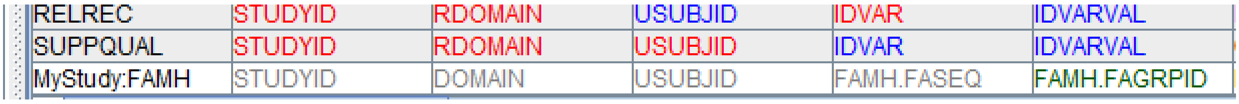
You may select to add a CDISC codelist to FATESTCD, depending on the type of findings area.
You can however also create and add your own codelist.

Asthma Findings About Test Code
Cardiovascular Findings About Test Code
COPD Findings About Test Code
Diabetic Kidney Disease Findings About Test Code
Duche
Ebola V
Malari
Multipl
Schizo
Tuberculosis Findings About Test Code
Vaccines Findings About Test Code

Codelist Name: COPD Findings About Test Code
Members:
CUMEXP - Cumulative Exposure
EPMLDNUM - Number of Mild Episodes
EPMODNUM - Number of Moderate Episodes
EPSEVNUM - Number of Severe Episodes

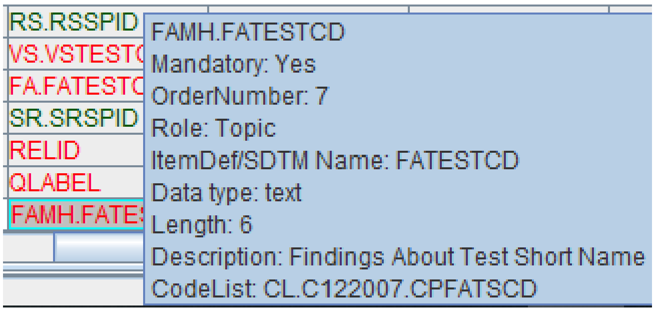
This allows to easier find a suitable codelist.
It is not mandatory to select an "—FATSCD" codelist, one can also not select anything (or click the Cancel button). One can then still later attach a codelist to both variables, and/or create a new one and attach it.
Also, when selecting an "—FATSCD" codelist, also the corresponding "—FATS" codelist will be loaded.

For example, when the "COPD Findings About Test Code" is selected, and OK is clicked, the FAMH instance is created:



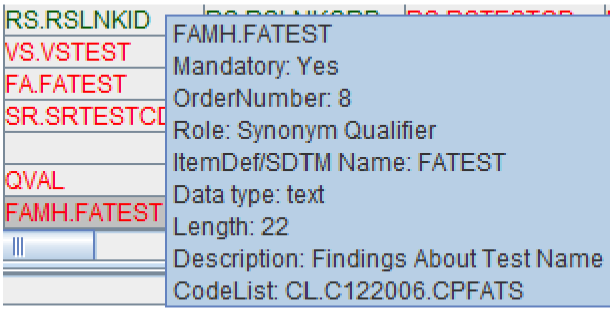
RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL
MyStudy:FAMH	STUDYID	DOMAIN	USUBJID	FAMH.FASEQ	FAMH.FAGRPID

And the "COPD Findings About Test Code" is attached to FATESTCD:



FAMH.FATESTCD
Mandatory: Yes
OrderNumber: 7
Role: Topic
ItemDef/SDTM Name: FATESTCD
Data type: text
Length: 6
Description: Findings About Test Short Name
CodeList: CL.C122007.CPFATSCD

And the corresponding codelist is attached to FATEST:



FAMH.FATEST
Mandatory: Yes
OrderNumber: 8
Role: Synonym Qualifier
ItemDef/SDTM Name: FATEST
Data type: text
Length: 22
Description: Findings About Test Name
CodeList: CL.C122006.CPFATS

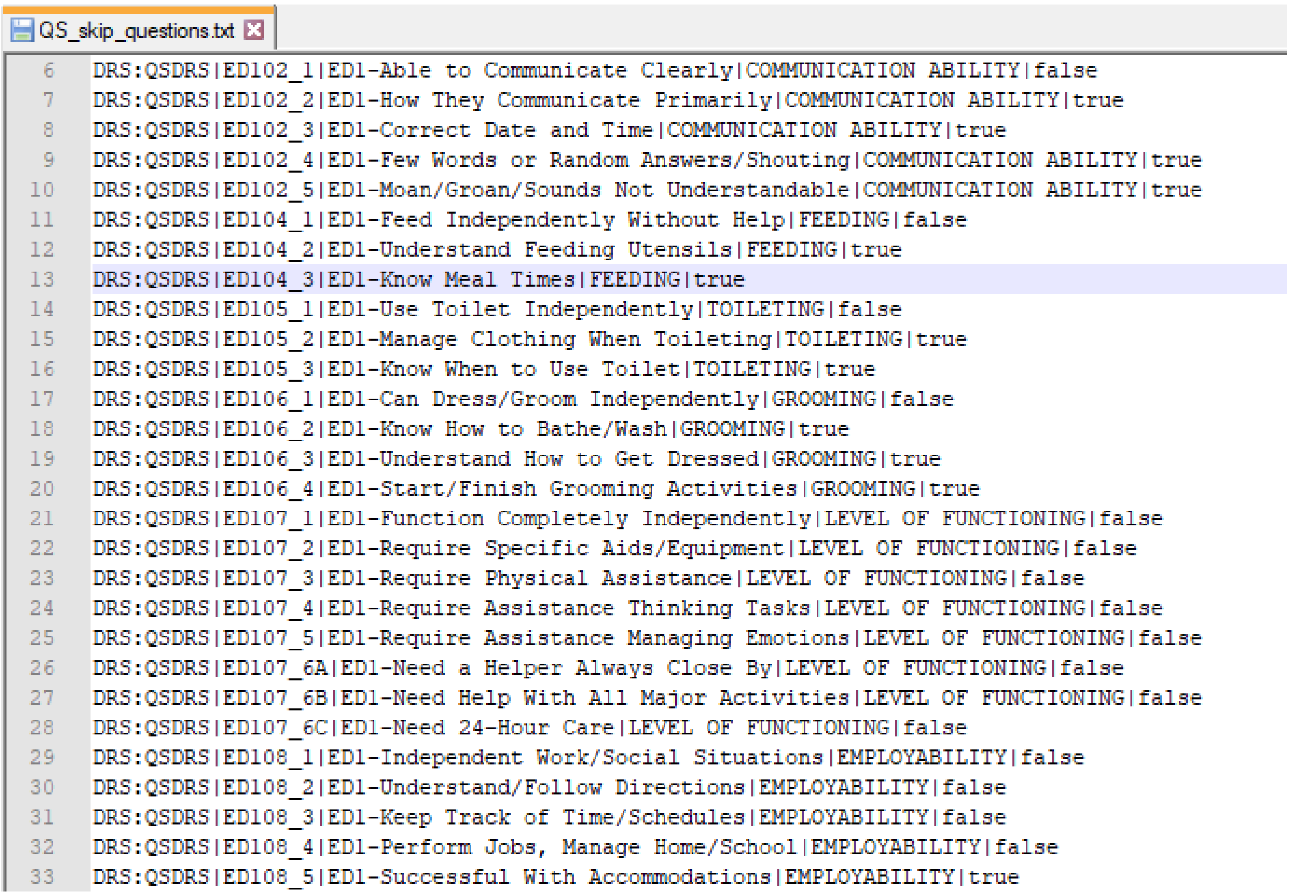
One can now start mapping as in the usual case, for example for all findings about medical history of COPD.

More details are found in the tutorial "**Working with the Findings About domain**"

Logically Skipped Items in SDTM-QS

In some cases, the FDA publishes rules for SDTM that override existing SDTM rules (although also the FDA can comment during public review periods) and sometimes even conflicts with them.
The newest example of this is the rule that for QS (Questionnaires) datasets, also data of logically skipped questions must be submitted. See the latest "Technical Conformance Guide".
This is unproblematic when such skipped items are collected and marked as such in the database, e.g. using a code "999". In most cases however (we checked with EDC and ePRO vendors), skipped questions are not explicitly collected and stored in the study database, as there is no data from them (logical, isn't it?), and because one can always do a lookup in the "skip rules" in order to see why a datapoint is missing. Such skip rules can easily be defined in the define.xml, but it looks as the FDA did not think about that.
So, this new rule forces SDTM generation tools to create records for non-collected data that usually are not in the database, and thus also not in the ODM export.
If the database however also contains records for skipped questions (such as "999" records), there is no problem at all.

In SDTM-ETL, we have solved this in an intelligent way. For the case that a questionnaire has "skip rules", such as the "Disability Rating Scale" questionnaire, for which also an annotated CRF exists, one can add a file "QS_skip_questions.txt" to the directory "QS_Skip_Questions". This file then contains the information whether an item can be skipped, and will be used to generate "NOT DONE" records for the skipped questions when that information is not explicitly available in the source database and ODM.
The SDTM-ETL distribution already contains such an example file for the "Disability Rating Scale". Here are some of the contents:

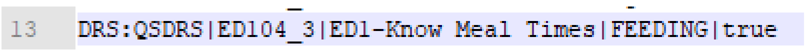


```
DRS:QSDRS|ED102_1|ED1-Able to Communicate Clearly|COMMUNICATION ABILITY|false
DRS:QSDRS|ED102_2|ED1-How They Communicate Primarily|COMMUNICATION ABILITY|true
DRS:QSDRS|ED102_3|ED1-Correct Date and Time|COMMUNICATION ABILITY|true
DRS:QSDRS|ED102_4|ED1-Few Words or Random Answers/Shouting|COMMUNICATION ABILITY|true
DRS:QSDRS|ED102_5|ED1-Moan/Groan/Sounds Not Understandable|COMMUNICATION ABILITY|true
DRS:QSDRS|ED104_1|ED1-Feed Independently Without Help|FEEDING|false
DRS:QSDRS|ED104_2|ED1-Understand Feeding Utensils|FEEDING|true
DRS:QSDRS|ED104_3|ED1-Know Meal Times|FEEDING|true
DRS:QSDRS|ED105_1|ED1-Use Toilet Independently|TOILETING|false
DRS:QSDRS|ED105_2|ED1-Manage Clothing When Toileting|TOILETING|true
DRS:QSDRS|ED105_3|ED1-Know When to Use Toilet|TOILETING|true
DRS:QSDRS|ED106_1|ED1-Can Dress/Groom Independently|GROOMING|false
DRS:QSDRS|ED106_2|ED1-Know How to Bathe/Wash|GROOMING|true
DRS:QSDRS|ED106_3|ED1-Understand How to Get Dressed|GROOMING|true
DRS:QSDRS|ED106_4|ED1-Start/Finish Grooming Activities|GROOMING|true
DRS:QSDRS|ED107_1|ED1-Function Completely Independently|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_2|ED1-Require Specific Aids/Equipment|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_3|ED1-Require Physical Assistance|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_4|ED1-Require Assistance Thinking Tasks|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_5|ED1-Require Assistance Managing Emotions|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_6A|ED1-Need a Helper Always Close By|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_6B|ED1-Need Help With All Major Activities|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED107_6C|ED1-Need 24-Hour Care|LEVEL OF FUNCTIONING|false
DRS:QSDRS|ED108_1|ED1-Independent Work/Social Situations|EMPLOYABILITY|false
DRS:QSDRS|ED108_2|ED1-Understand/Follow Directions|EMPLOYABILITY|false
DRS:QSDRS|ED108_3|ED1-Keep Track of Time/Schedules|EMPLOYABILITY|false
DRS:QSDRS|ED108_4|ED1-Perform Jobs, Manage Home/School|EMPLOYABILITY|false
DRS:QSDRS|ED108_5|ED1-Successful With Accommodations|EMPLOYABILITY|true
```

Fields in the file are separated by a vertical bar "|". The fields are:

- Field 1: The QS dataset identifier
- Field 2: The question identifier (OID of the ItemDef in the ODM file – for populating QSTESTCD)
- Field 3: The question label (for populating QSTEST)
- Field 4: The category (for filling QSCAT)
- Field 5: Boolean value whether the question can be skipped or not: when the value is "false", no "NOT DONE" record will ever be created for that item, as it could never be "logically skipped".

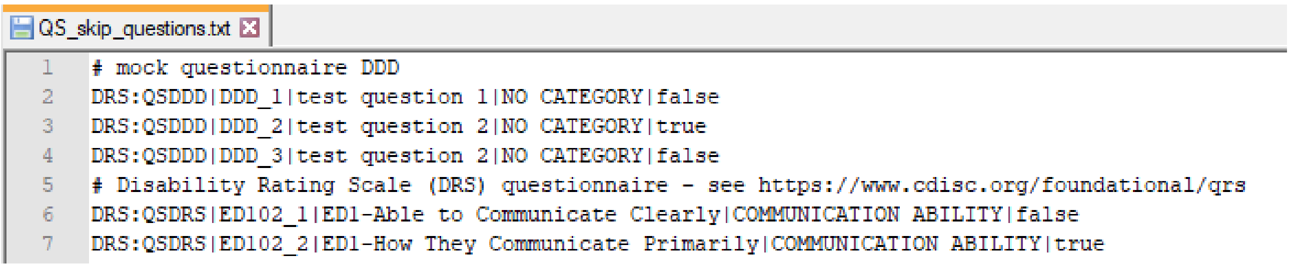
For example, the line:



```
DRS:QSDRS|ED104_3|ED1-Know Meal Times|FEEDING|true
```

Indicates that for the dataset with identifier "DRS:QSDRS" (i.e. the QSDRS dataset in study "DRS") has an item "ED104_3" (the test code), with test name "ED1-Know Meal Times", with the category (QSCAT) "FEEDING" is an item that can be skipped under certain circumstances.
If the software then detects that for a certain subject/visit combination, there is no data for item "ED104_3", it will generated a record with QSORRES empty, QSSTAT="NOT DONE" and QSREASND="LOGICALLY SKIPPED ITEM".

This file can contain information for more than one QS datasets. For example:

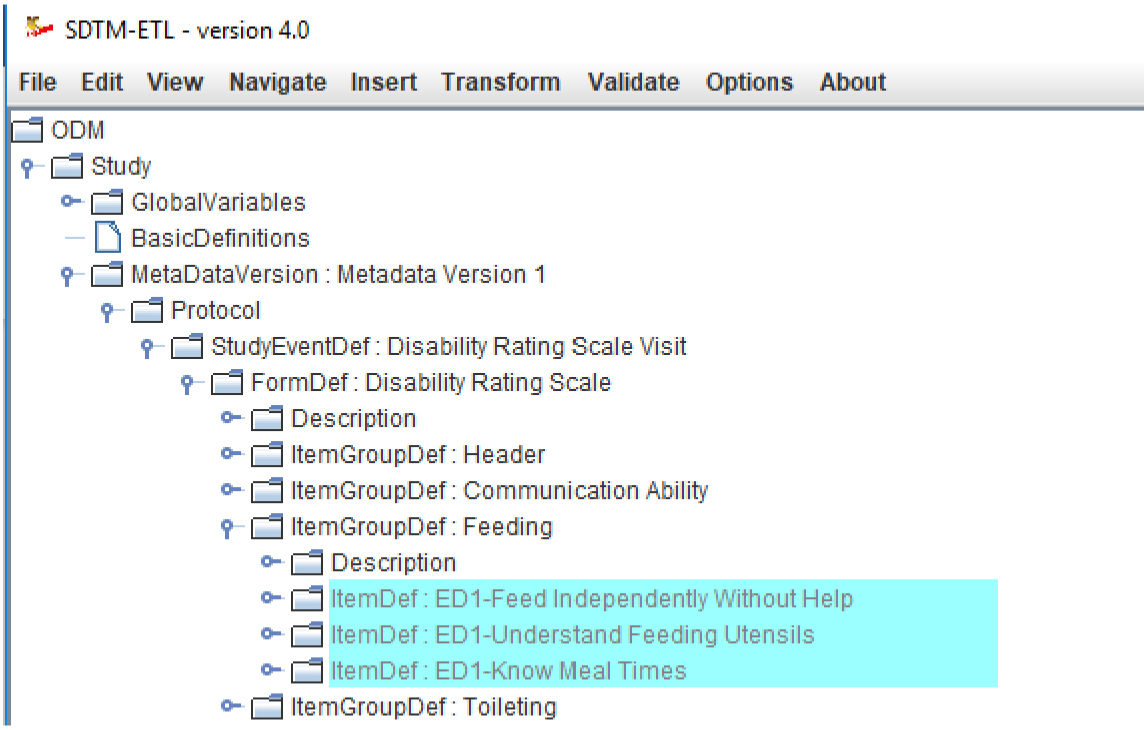


```
# mock questionnaire DDD
DRS:QSDDD|DDD_1|test question 1|NO CATEGORY|false
DRS:QSDDD|DDD_2|test question 2|NO CATEGORY|true
DRS:QSDDD|DDD_3|test question 2|NO CATEGORY|false
# Disability Rating Scale (DRS) questionnaire - see https://www.cdisc.org/foundational/qrs
DRS:QSDRS|ED102_1|ED1-Able to Communicate Clearly|COMMUNICATION ABILITY|false
DRS:QSDRS|ED102_2|ED1-How They Communicate Primarily|COMMUNICATION ABILITY|true
```

As one may already guess, lines starting with a "#" are comment files and will be ignored by the software.

If the software finds more than one file, or none at all, in the directory "QS_Skip_Questions", it will ask which one to use, and/or allow the user to select such a file.

In the SDTM-ETL, the ODM file with metadata for the "Disability Rating Scale" questionnaire, when loaded looks like[2]:

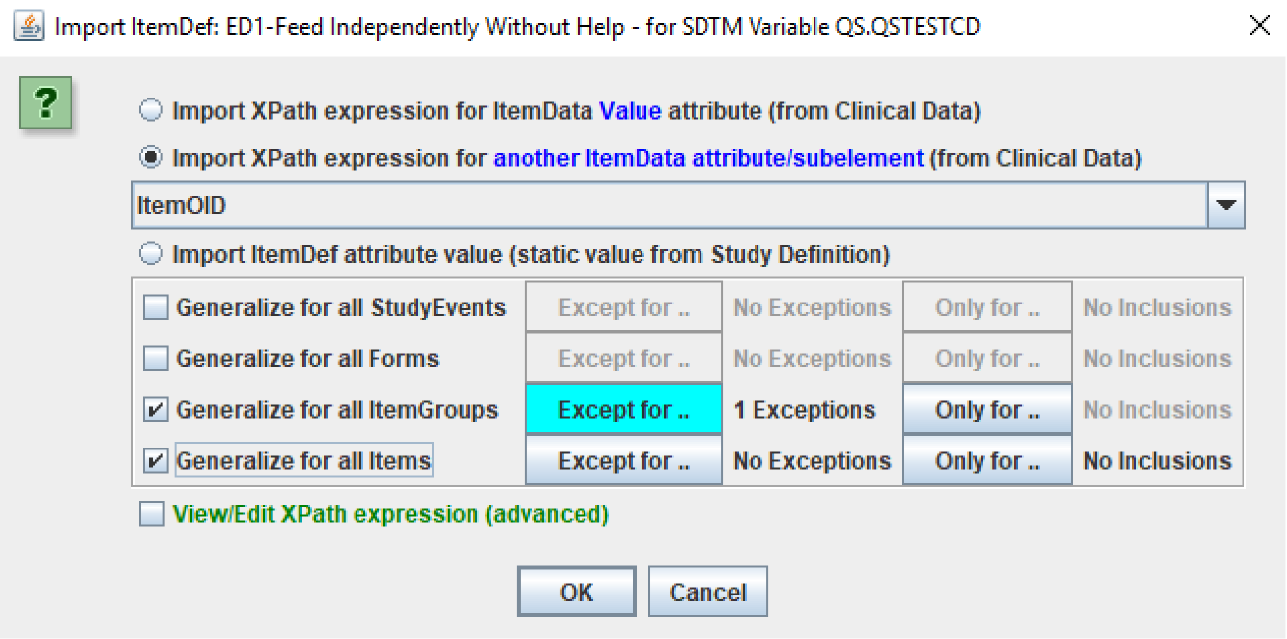

SDTM-ETL - version 4.0
File
Edit
View
Navigate
Insert
Transform
Validate
Options
About
ODM
Study
GlobalVariables
BasicDefinitions
MetaDataVersion : Metadata Version 1
Protocol
StudyEventDef : Disability Rating Scale Visit
FormDef : Disability Rating Scale
Description
ItemGroupDef : Header
ItemGroupDef : Communication Ability
ItemGroupDef : Feeding
Description
ItemDef : ED1-Feed Independently Without Help
ItemDef : ED1-Understand Feeding Utensils
ItemDef : ED1-Know Meal Times
ItemGroupDef : Toileting

The SDTM-ETL script for QSTESTCD looks like:

```
# Mapping using ODM element ItemData with ItemOID ED102_1 - value from attribute ItemOID
# Generalized for all ItemGroups within the Form
# Except for: IG.HEADER
# Generalized for all Items within the ItemGroup
$QS.QSTESTCD =
xpath(/StudyEventData[@StudyEventOID='SE.DRS']/FormData[@FormOID='FO.DRS']/ItemGroupData[not(@ItemGroupOID='IG.HEADER')]/ItemData/@ItemOID);
```

Stating an iteration over all items in the form, except for the items in the "header". It was automatically created after a simple drag and drop and using the wizard:

[2] A copy of the "Disability Rating Scale" ODM file and the corresponding SDTM-ETL define.xml file that were used for testing this new feature, can be obtained upon request.



Import ItemDef: ED1-Feed Independently Without Help - for SDTM Variable QS.QSTESTCD

○ Import XPath expression for ItemData Value attribute (from Clinical Data)

● Import XPath expression for another ItemData attribute/subelement (from Clinical Data)

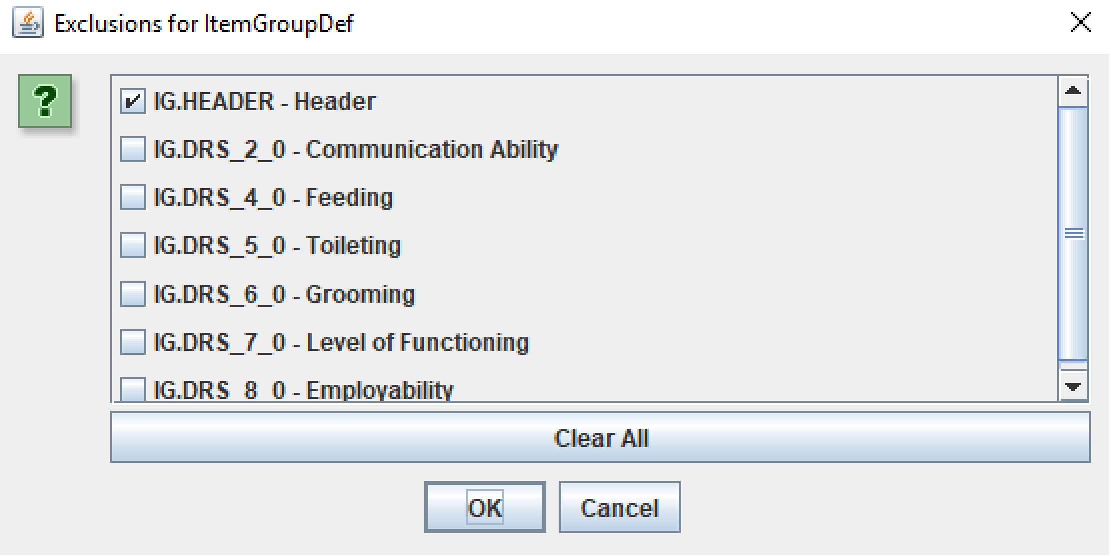
ItemOID

○ Import ItemDef attribute value (static value from Study Definition)

□ Generalize for all StudyEvents	Except for ..	No Exceptions	Only for ..	No Inclusions
□ Generalize for all Forms	Except for ..	No Exceptions	Only for ..	No Inclusions
☑ Generalize for all ItemGroups	Except for ..	1 Exceptions	Only for ..	No Inclusions
☑ Generalize for all Items	Except for ..	No Exceptions	Only for ..	No Inclusions

□ View/Edit XPath expression (advanced)

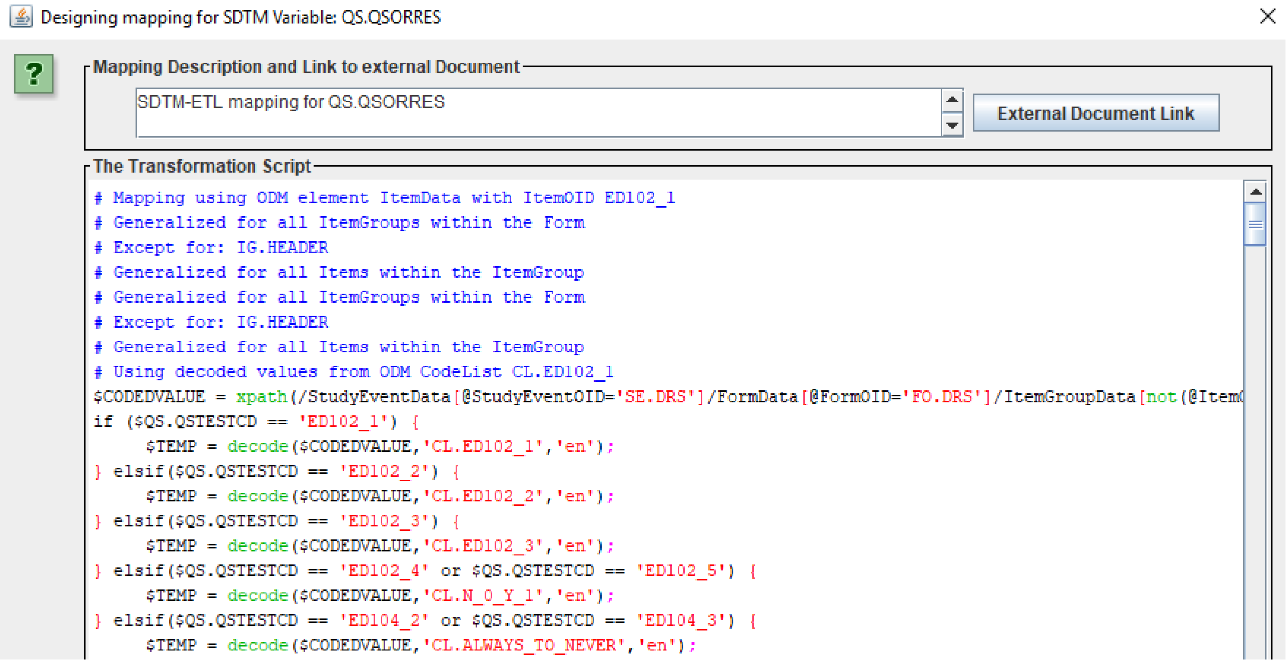
With the one exception generated by clicking "Except for …" in "Generalize for all ItemGroups":



Exclusions for ItemGroupDef

☑ IG.HEADER - Header
□ IG.DRS_2_0 - Communication Ability
□ IG.DRS_4_0 - Feeding
□ IG.DRS_5_0 - Toileting
□ IG.DRS_6_0 - Grooming
□ IG.DRS_7_0 - Level of Functioning
□ IG.DRS_8_0 - Employability

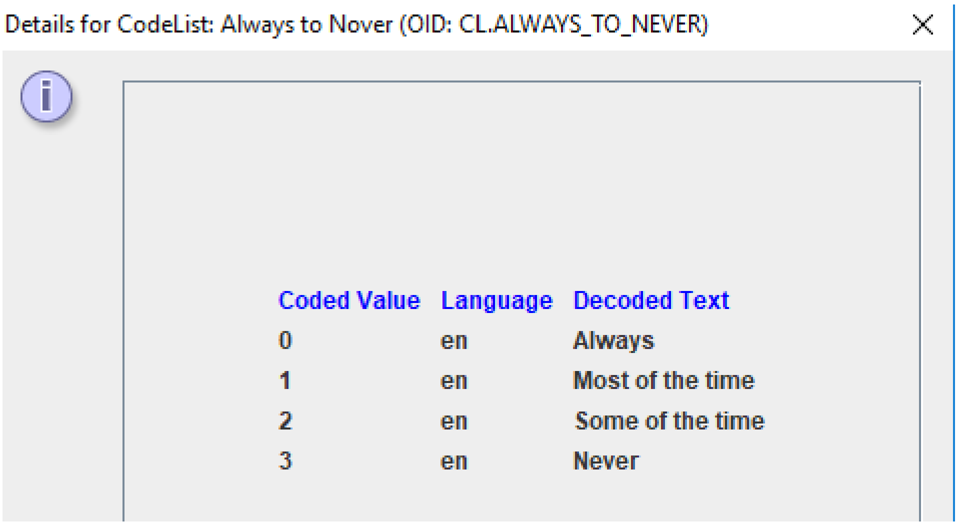
The script for "QSORRES" was generated in a very similar way and looks like:


Designing mapping for SDTM Variable: QS.QSORRES
?
Mapping Description and Link to external Document
External Document Link
The Transformation Script

SDTM-ETL mapping for QS.QSORRES

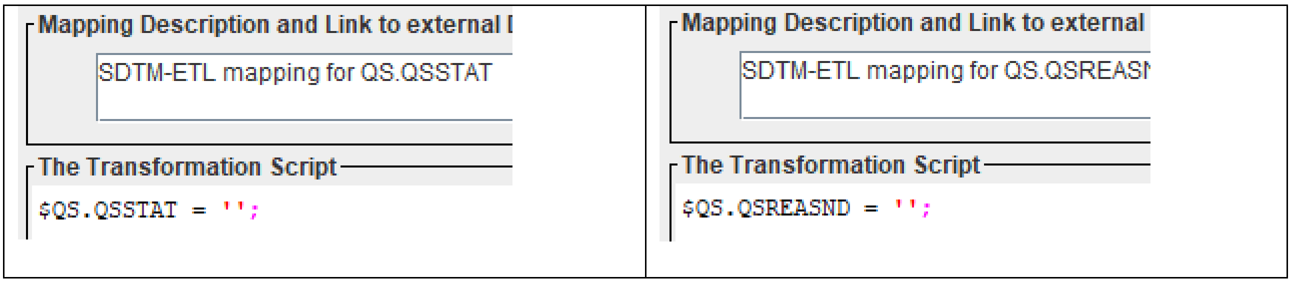
```
# Mapping using ODM element ItemData with ItemOID ED102_1
# Generalized for all ItemGroups within the Form
# Except for: IG.HEADER
# Generalized for all Items within the ItemGroup
# Generalized for all ItemGroups within the Form
# Except for: IG.HEADER
# Generalized for all Items within the ItemGroup
# Using decoded values from ODM CodeList CL.ED102_1
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='SE.DRS']/FormData[@FormOID='FO.DRS']/ItemGroupData[not(@Item
if ($QS.QSTESTCD == 'ED102_1') {
     $TEMP = decode($CODEDVALUE,'CL.ED102_1','en');
} elsif($QS.QSTESTCD == 'ED102_2') {
     $TEMP = decode($CODEDVALUE,'CL.ED102_2','en');
} elsif($QS.QSTESTCD == 'ED102_3') {
     $TEMP = decode($CODEDVALUE,'CL.ED102_3','en');
} elsif($QS.QSTESTCD == 'ED102_4' or $QS.QSTESTCD == 'ED102_5') {
     $TEMP = decode($CODEDVALUE,'CL.N_0_Y_1','en');
} elsif($QS.QSTESTCD == 'ED104_2' or $QS.QSTESTCD == 'ED104_3') {
     $TEMP = decode($CODEDVALUE,'CL.ALWAYS_TO_NEVER','en');
```

It uses the "decode()" function as the answers in the database are stored as numeric values, like "0" for "Always", "1" for "Most of the time", "2" for "Some of the time" and "3" for "Never". Storing the answers in the database as numeric values makes sense, as these numbers represent a score, from which the total score is calculated.
This can be easily visualized using the menu "View – Item Associated CodeList":


Details for CodeList: Always to Nover (OID: CL.ALWAYS_TO_NEVER)

Coded Value	Language	Decoded Text
0	en	Always
1	en	Most of the time
2	en	Some of the time
3	en	Never

We also need take care that there is a "dummy" mapping specified in QSSTAT and QSREASND, as these will be automatically populated in a post-mapping run, i.e. this run will override the values in QSSTAT and QSREASND:


Mapping Description and Link to external I
The Transformation Script
Mapping Description and Link to external
The Transformation Script

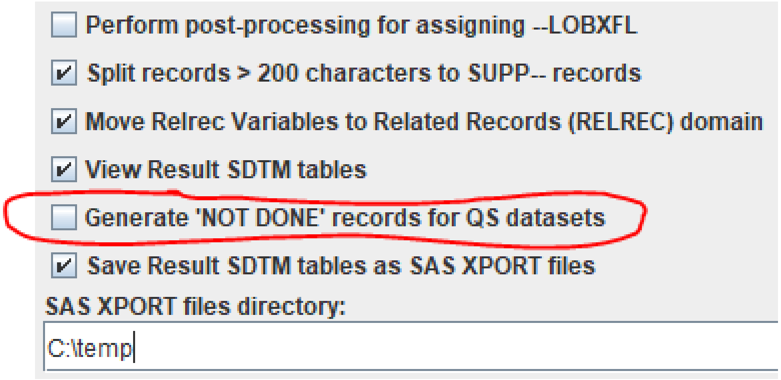
SDTM-ETL mapping for QS.QSSTAT

```
$QS.QSSTAT = '';
```

SDTM-ETL mapping for QS.QSREASN

```
$QS.QSREASND = '';
```

When we now run "Transform – Generate Transformation Code" followed by "Execute Transformation Code", the system detects that there is a study-specific QS dataset instance defined, and displays an additional checkbox "Generate 'NOT DONE' records for QS datasets":



□ Perform post-processing for assigning --LOBXFL

☑ Split records > 200 characters to SUPP-- records

☑ Move Relrec Variables to Related Records (RELREC) domain

☑ View Result SDTM tables

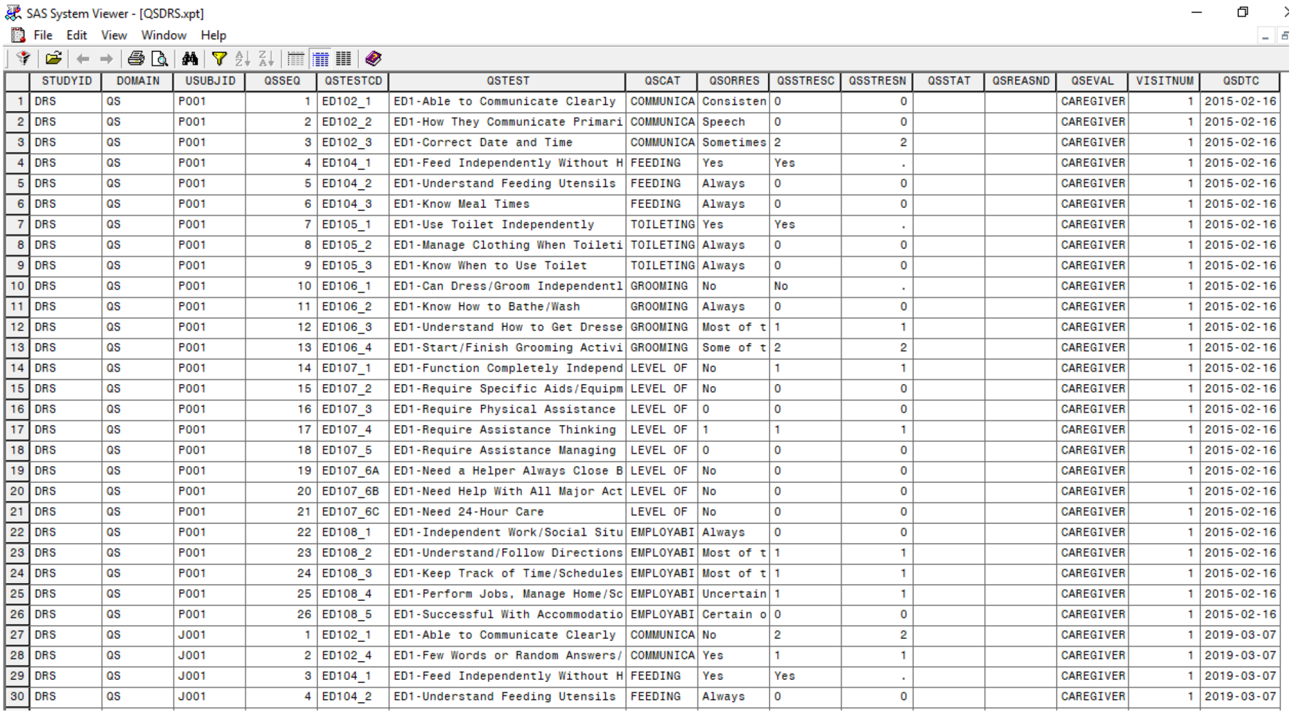
□ Generate 'NOT DONE' records for QS datasets

☑ Save Result SDTM tables as SAS XPORT files

SAS XPORT files directory:

C:\temp

If we leave this checkbox unchecked, nothing special will happen, and our QSDRS dataset will look like:

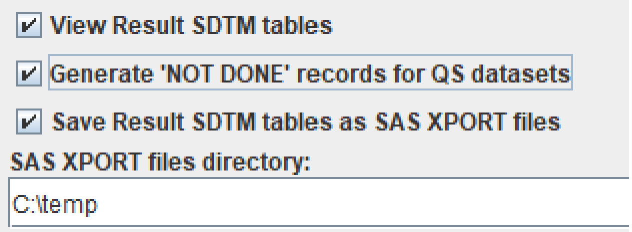


SAS System Viewer - [QSDRS.xpt]

	STUDYID	DOMAIN	USUBJID	QSSEQ	QSTESTCD	QSTEST	QSCAT	QSORRES	QSSTRESC	QSSTRESN	QSSTAT	QSREASND	QSEVAL	VISITNUM	QSDTC
1	DRS	QS	P001	1	ED102_1	ED1-Able to Communicate Clearly	COMMUNICA	Consisten	0	0			CAREGIVER	1	2015-02-16
2	DRS	QS	P001	2	ED102_2	ED1-How They Communicate Primari	COMMUNICA	Speech	0	0			CAREGIVER	1	2015-02-16
3	DRS	QS	P001	3	ED102_3	ED1-Correct Date and Time	COMMUNICA	Sometimes	2	2			CAREGIVER	1	2015-02-16
4	DRS	QS	P001	4	ED104_1	ED1-Feed Independently Without H	FEEDING	Yes	Yes	.			CAREGIVER	1	2015-02-16
5	DRS	QS	P001	5	ED104_2	ED1-Understand Feeding Utensils	FEEDING	Always	0	0			CAREGIVER	1	2015-02-16
6	DRS	QS	P001	6	ED104_3	ED1-Know Meal Times	FEEDING	Always	0	0			CAREGIVER	1	2015-02-16
7	DRS	QS	P001	7	ED105_1	ED1-Use Toilet Independently	TOILETING	Yes	Yes	.			CAREGIVER	1	2015-02-16
8	DRS	QS	P001	8	ED105_2	ED1-Manage Clothing When Toileti	TOILETING	Always	0	0			CAREGIVER	1	2015-02-16
9	DRS	QS	P001	9	ED105_3	ED1-Know When to Use Toilet	TOILETING	Always	0	0			CAREGIVER	1	2015-02-16
10	DRS	QS	P001	10	ED106_1	ED1-Can Dress/Groom Independentl	GROOMING	No	No	.			CAREGIVER	1	2015-02-16
11	DRS	QS	P001	11	ED106_2	ED1-Know How to Bathe/Wash	GROOMING	Always	0	0			CAREGIVER	1	2015-02-16
12	DRS	QS	P001	12	ED106_3	ED1-Understand How to Get Dresse	GROOMING	Most of t	1	1			CAREGIVER	1	2015-02-16
13	DRS	QS	P001	13	ED106_4	ED1-Start/Finish Grooming Activi	GROOMING	Some of t	2	2			CAREGIVER	1	2015-02-16
14	DRS	QS	P001	14	ED107_1	ED1-Function Completely Independ	LEVEL OF	No	1	1			CAREGIVER	1	2015-02-16
15	DRS	QS	P001	15	ED107_2	ED1-Require Specific Aids/Equipm	LEVEL OF	No	0	0			CAREGIVER	1	2015-02-16
16	DRS	QS	P001	16	ED107_3	ED1-Require Physical Assistance	LEVEL OF	0	0	0			CAREGIVER	1	2015-02-16
17	DRS	QS	P001	17	ED107_4	ED1-Require Assistance Thinking	LEVEL OF	1	1	1			CAREGIVER	1	2015-02-16
18	DRS	QS	P001	18	ED107_5	ED1-Require Assistance Managing	LEVEL OF	0	0	0			CAREGIVER	1	2015-02-16
19	DRS	QS	P001	19	ED107_6A	ED1-Need a Helper Always Close B	LEVEL OF	No	0	0			CAREGIVER	1	2015-02-16
20	DRS	QS	P001	20	ED107_6B	ED1-Need Help With All Major Act	LEVEL OF	No	0	0			CAREGIVER	1	2015-02-16
21	DRS	QS	P001	21	ED107_6C	ED1-Need 24-Hour Care	LEVEL OF	No	0	0			CAREGIVER	1	2015-02-16
22	DRS	QS	P001	22	ED108_1	ED1-Independent Work/Social Situ	EMPLOYABI	Always	0	0			CAREGIVER	1	2015-02-16
23	DRS	QS	P001	23	ED108_2	ED1-Understand/Follow Directions	EMPLOYABI	Most of t	1	1			CAREGIVER	1	2015-02-16
24	DRS	QS	P001	24	ED108_3	ED1-Keep Track of Time/Schedules	EMPLOYABI	Most of t	1	1			CAREGIVER	1	2015-02-16
25	DRS	QS	P001	25	ED108_4	ED1-Perform Jobs, Manage Home/Sc	EMPLOYABI	Uncertain	1	1			CAREGIVER	1	2015-02-16
26	DRS	QS	P001	26	ED108_5	ED1-Successful With Accommodatio	EMPLOYABI	Certain o	0	0			CAREGIVER	1	2015-02-16
27	DRS	QS	J001	1	ED102_1	ED1-Able to Communicate Clearly	COMMUNICA	No	2	2			CAREGIVER	1	2019-03-07
28	DRS	QS	J001	2	ED102_4	ED1-Few Words or Random Answers/	COMMUNICA	Yes	1	1			CAREGIVER	1	2019-03-07
29	DRS	QS	J001	3	ED104_1	ED1-Feed Independently Without H	FEEDING	Yes	Yes	.			CAREGIVER	1	2019-03-07
30	DRS	QS	J001	4	ED104_2	ED1-Understand Feeding Utensils	FEEDING	Always	0	0			CAREGIVER	1	2019-03-07

QSSTRESN (and thus also QSSTRESC) contain the numeric (coded) values from the source database, whereas QSORRES contains the text value of the answer. This is also as described in the CDISC document. Remark that "skipped questions" according to DRS rules are … skipped, and do not appear in the dataset.

When however the checkbox "Generate 'NOT DONE' records for QS datasets" is checked:



☑ View Result SDTM tables

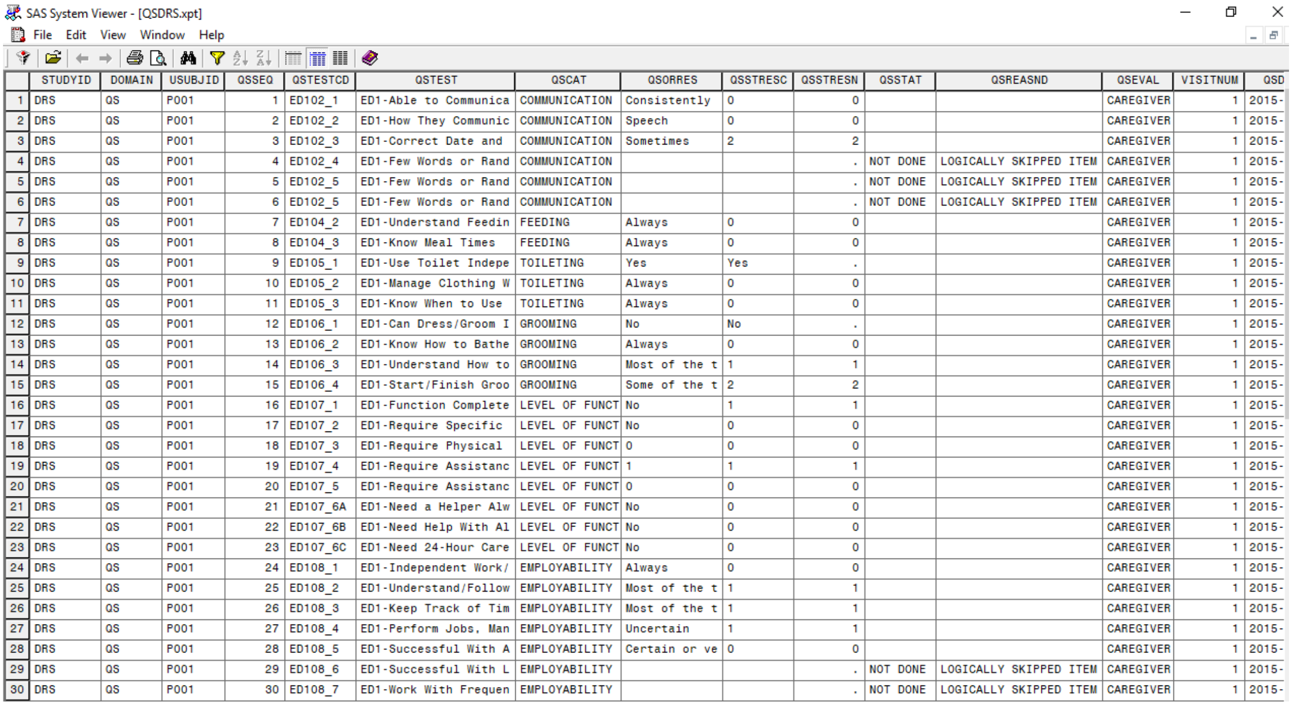
☑ Generate 'NOT DONE' records for QS datasets

☑ Save Result SDTM tables as SAS XPORT files

SAS XPORT files directory:

C:\temp

The result is:



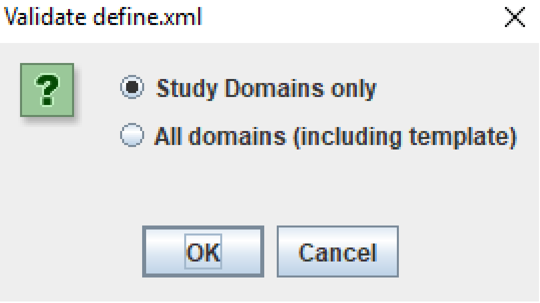
	STUDYID	DOMAIN	USUBJID	QSSEQ	QSTESTCD	QSTEST	QSCAT	QSORRES	QSSTRESC	QSSTRESN	QSSTAT	QSREASND	QSEVAL	VISITNUM	QSD
1	DRS	QS	P001	1	ED102_1	ED1-Able to Communica	COMMUNICATION	Consistently	0	0			CAREGIVER	1	2015-
2	DRS	QS	P001	2	ED102_2	ED1-How They Communic	COMMUNICATION	Speech	0	0			CAREGIVER	1	2015-
3	DRS	QS	P001	3	ED102_3	ED1-Correct Date and	COMMUNICATION	Sometimes	2	2			CAREGIVER	1	2015-
4	DRS	QS	P001	4	ED102_4	ED1-Few Words or Rand	COMMUNICATION			.	NOT DONE	LOGICALLY SKIPPED ITEM	CAREGIVER	1	2015-
5	DRS	QS	P001	5	ED102_5	ED1-Few Words or Rand	COMMUNICATION			.	NOT DONE	LOGICALLY SKIPPED ITEM	CAREGIVER	1	2015-
6	DRS	QS	P001	6	ED102_5	ED1-Few Words or Rand	COMMUNICATION			.	NOT DONE	LOGICALLY SKIPPED ITEM	CAREGIVER	1	2015-
7	DRS	QS	P001	7	ED104_2	ED1-Understand Feedin	FEEDING	Always	0	0			CAREGIVER	1	2015-
8	DRS	QS	P001	8	ED104_3	ED1-Know Meal Times	FEEDING	Always	0	0			CAREGIVER	1	2015-
9	DRS	QS	P001	9	ED105_1	ED1-Use Toilet Indepe	TOILETING	Yes	Yes	.			CAREGIVER	1	2015-
10	DRS	QS	P001	10	ED105_2	ED1-Manage Clothing W	TOILETING	Always	0	0			CAREGIVER	1	2015-
11	DRS	QS	P001	11	ED105_3	ED1-Know When to Use	TOILETING	Always	0	0			CAREGIVER	1	2015-
12	DRS	QS	P001	12	ED106_1	ED1-Can Dress/Groom I	GROOMING	No	No	.			CAREGIVER	1	2015-
13	DRS	QS	P001	13	ED106_2	ED1-Know How to Bathe	GROOMING	Always	0	0			CAREGIVER	1	2015-
14	DRS	QS	P001	14	ED106_3	ED1-Understand How to	GROOMING	Most of the t	1	1			CAREGIVER	1	2015-
15	DRS	QS	P001	15	ED106_4	ED1-Start/Finish Groo	GROOMING	Some of the t	2	2			CAREGIVER	1	2015-
16	DRS	QS	P001	16	ED107_1	ED1-Function Complete	LEVEL OF FUNCT	No	1	1			CAREGIVER	1	2015-
17	DRS	QS	P001	17	ED107_2	ED1-Require Specific	LEVEL OF FUNCT	No	0	0			CAREGIVER	1	2015-
18	DRS	QS	P001	18	ED107_3	ED1-Require Physical	LEVEL OF FUNCT	0	0	0			CAREGIVER	1	2015-
19	DRS	QS	P001	19	ED107_4	ED1-Require Assistanc	LEVEL OF FUNCT	1	1	1			CAREGIVER	1	2015-
20	DRS	QS	P001	20	ED107_5	ED1-Require Assistanc	LEVEL OF FUNCT	0	0	0			CAREGIVER	1	2015-
21	DRS	QS	P001	21	ED107_6A	ED1-Need a Helper Alw	LEVEL OF FUNCT	No	0	0			CAREGIVER	1	2015-
22	DRS	QS	P001	22	ED107_6B	ED1-Need Help With Al	LEVEL OF FUNCT	No	0	0			CAREGIVER	1	2015-
23	DRS	QS	P001	23	ED107_6C	ED1-Need 24-Hour Care	LEVEL OF FUNCT	No	0	0			CAREGIVER	1	2015-
24	DRS	QS	P001	24	ED108_1	ED1-Independent Work/	EMPLOYABILITY	Always	0	0			CAREGIVER	1	2015-
25	DRS	QS	P001	25	ED108_2	ED1-Understand/Follow	EMPLOYABILITY	Most of the t	1	1			CAREGIVER	1	2015-
26	DRS	QS	P001	26	ED108_3	ED1-Keep Track of Tim	EMPLOYABILITY	Most of the t	1	1			CAREGIVER	1	2015-
27	DRS	QS	P001	27	ED108_4	ED1-Perform Jobs, Man	EMPLOYABILITY	Uncertain	1	1			CAREGIVER	1	2015-
28	DRS	QS	P001	28	ED108_5	ED1-Successful With A	EMPLOYABILITY	Certain or ve	0	0			CAREGIVER	1	2015-
29	DRS	QS	P001	29	ED108_6	ED1-Successful With L	EMPLOYABILITY			.	NOT DONE	LOGICALLY SKIPPED ITEM	CAREGIVER	1	2015-
30	DRS	QS	P001	30	ED108_7	ED1-Work With Frequen	EMPLOYABILITY			.	NOT DONE	LOGICALLY SKIPPED ITEM	CAREGIVER	1	2015-

We now see that there are "NOT DONE" records for ED102_4 to ED102_6 although the original ODM dataset does not contain any information about these "skipped questions". These records have been generated "post-mapping" from the information in the file "QS_skip_questions.txt".

Again, if your dataset also contains information about questions that were not asked or were skipped (e.g. as "999" records), all this is not necessary.

Validation of "Study-specific" domains only

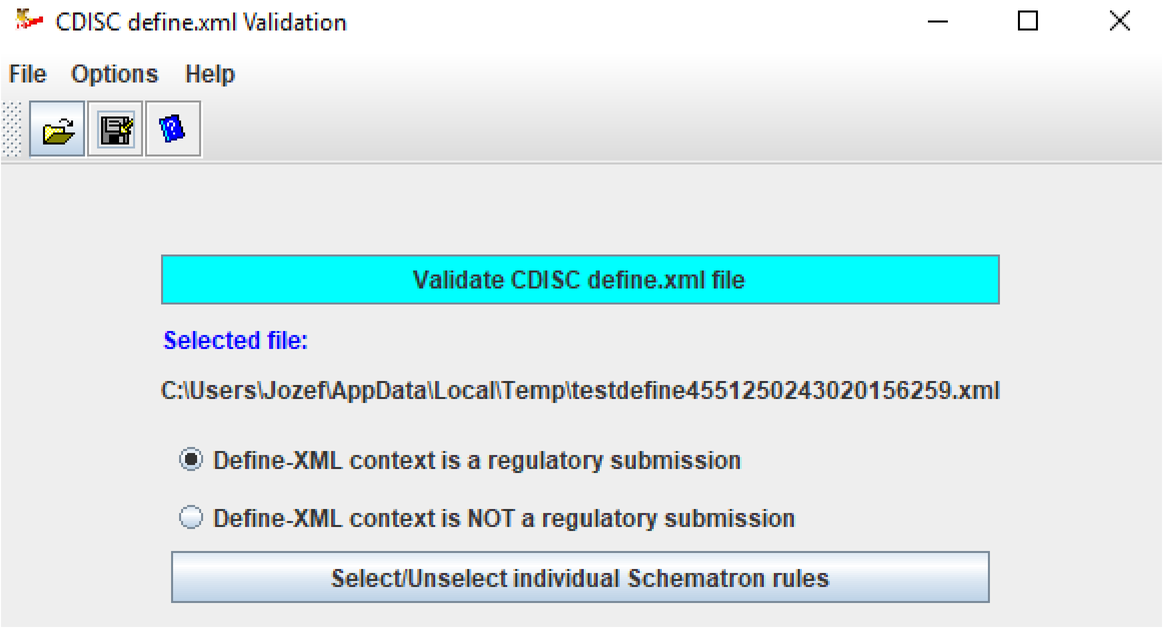
With some new requirements on the define.xml, the template "define.xml" does not always contain the necessary information to go through validation without warnings. Therefore, we introduced the feature that when validation is requested, the user can choose between validating the "study-specific instances" of the domains/datasets (i.e. the non-template rows) and validating everything. In most cases, the user will want to validate the "study-specific domains" only:


Validate define.xml

● Study Domains only

○ All domains (including template)

It is then asked whether the define.xml is in the context of a regulatory submission. The reason for this is that the rules for define.xml are slightly different when it is used for a regulatory submission (e.g. SASName is required):



CDISC define.xml Validation
File
Options
Help
Validate CDISC define.xml file
Select/Unselect individual Schematron rules

Selected file:

C:\Users\Jozef\AppData\Local\Temp\testdefine4551250243020156259.xml

● Define-XML context is a regulatory submission

○ Define-XML context is NOT a regulatory submission

Most of the rules for define.xml are expression as so-called Schematron rules. Schematron is an open, international standard for validation of XML files developed by the World Wide Web Consortium (w3c). The Schematron rules for define.xml have been developed by some of the members (including ourselves) of the CDISC define.xml development team.
Using the button "Select/Unselect individual Schematron rules" allows to select / unselect individual rules for define.xml.

Non-Standard variables and domains in define.xml 2.1

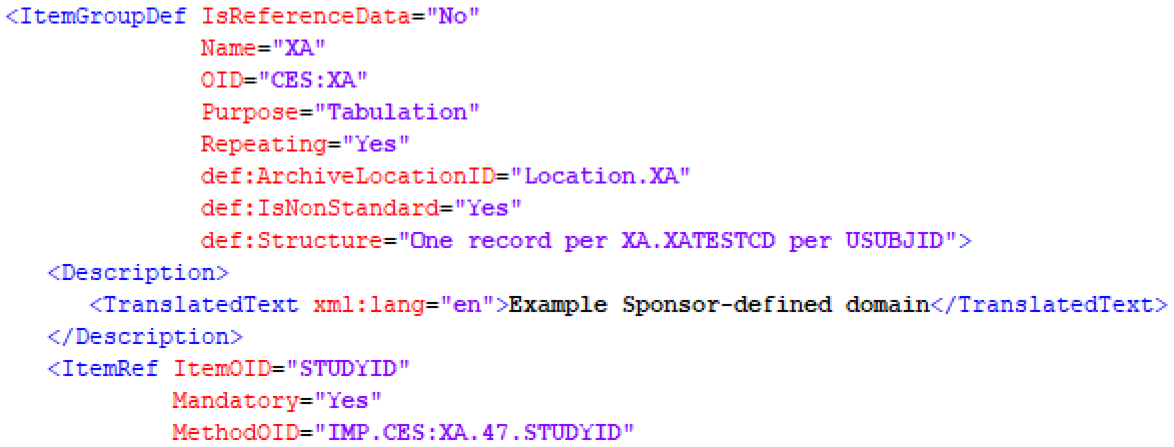
In define.xml 2.0, there was no standardized mechanism to flag non-standard variables (NSVs). So, in SDTM-ETL 3.2 and earlier, we marked them with "Role=SUPPQUAL". This information was then used to color them differently in the SDTM/SEND table on the right side of the screen, and to "split them off" to a SUPPxx dataset at execution time (when the datasets are generated) when desired by the user, and to generate a "submission-ready" define.xml including the SUPPxx dataset descriptions.
Also, define.xml did not have a mechanism to mark sponsor-defined domains and datasets. By convention, the two-character name of such domains and datasets started with either "X", "Y" or "Z".

In Define-XML 2.1, a new attribute was added to the "ItemRef" and "ItemGroupDef" elements: the "def:IsNonStandard" attribute. So, for example, the NSV "Completers Population Flag" (COMPLT) in the "Demographics" domain is defined by:

```
<ItemRef ItemOID="DM.DMDY" Mandatory="No" OrderNumber="28" Role="Timing"/>
<ItemRef ItemOID="DM.COMPLT"
         Mandatory="No"
         OrderNumber="29"
         Role="Record Qualifier"
         def:IsNonStandard="Yes"/>
<def:Class Name="SPECIAL PURPOSE"/>
```

This has the advantage that the user can now assign the value for the "Role" himself.

Similarly, for a sponsor defined domain/dataset:

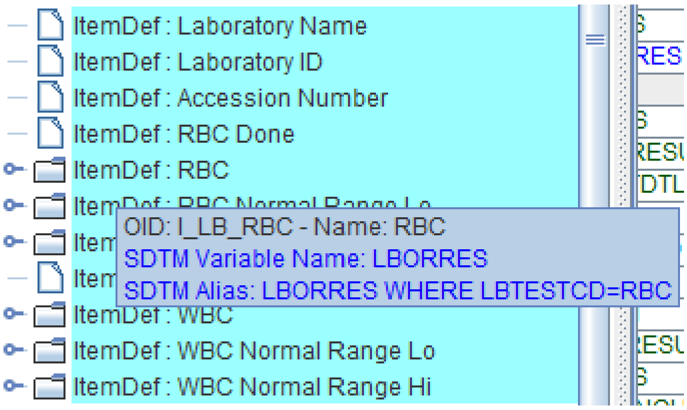


```
<ItemGroupDef IsReferenceData="No"
              Name="XA"
              OID="CES:XA"
              Purpose="Tabulation"
              Repeating="Yes"
              def:ArchiveLocationID="Location.XA"
              def:IsNonStandard="Yes"
              def:Structure="One record per XA.XATESTCD per USUBJID">
   <Description>
      <TranslatedText xml:lang="en">Example Sponsor-defined domain</TranslatedText>
   </Description>
   <ItemRef ItemOID="STUDYID"
            Mandatory="Yes"
            MethodOID="IMP.CES:XA.47.STUDYID"
```

This is all done in the background by the software, there is nothing special or new the user has to do.

Even better exploitation of SDTM (and CDASH) annotations in the ODM
(as of December 2021)

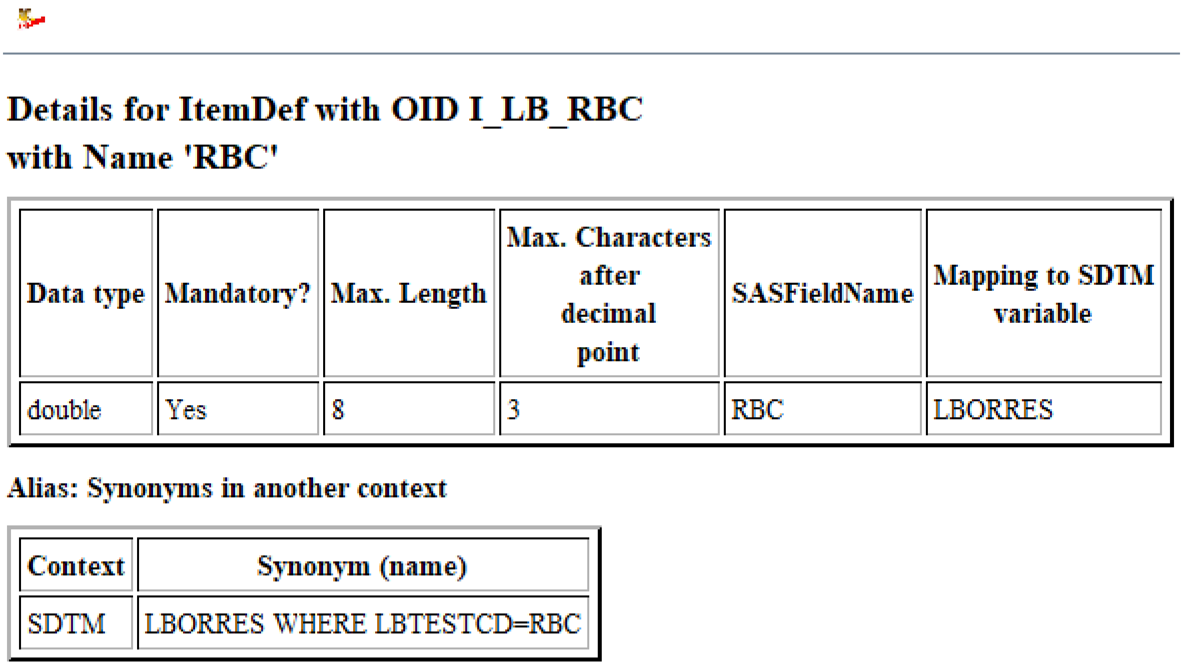
SDTM-ETL had already great features for exploiting SDTM (and CDASH) annotations in the ODM. Items that have such an annotation are displayed with a cyan background color, and upon pointing the mouse, the suggested SDTM variable is displayed as a tooltip:



ItemDef : Laboratory Name
ItemDef : Laboratory ID
ItemDef : Accession Number
ItemDef : RBC Done
ItemDef : RBC
ItemDef : RBC Normal Range Lo
Item
Item
ItemDef : WBC
ItemDef : WBC Normal Range Lo
ItemDef : WBC Normal Range Hi

OID: I_LB_RBC - Name: RBC
SDTM Variable Name: LBORRES
SDTM Alias: LBORRES WHERE LBTESTCD=RBC

This information is of course also displayed when using the menu “View - Item Details”:



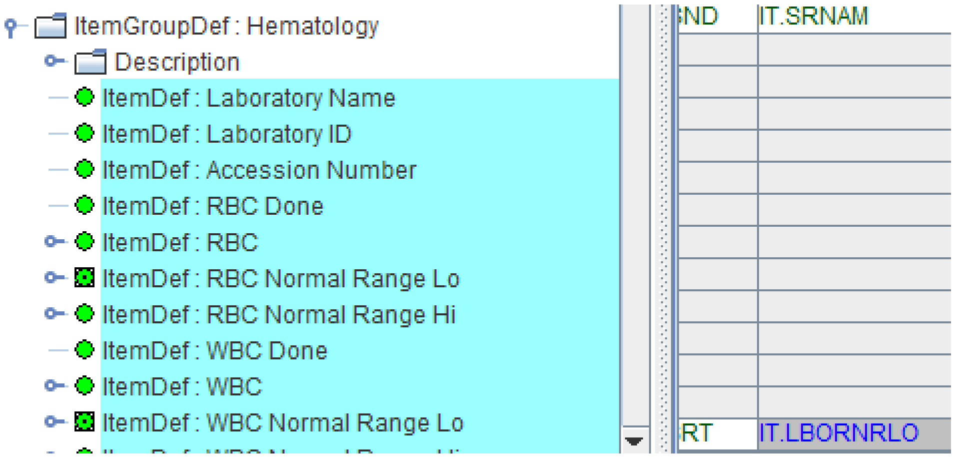
Details for ItemDef with OID I_LB_RBC with Name 'RBC'

Data type	Mandatory?	Max. Length	Max. Characters after decimal point	SASFieldName	Mapping to SDTM variable
double	Yes	8	3	RBC	LBORRES

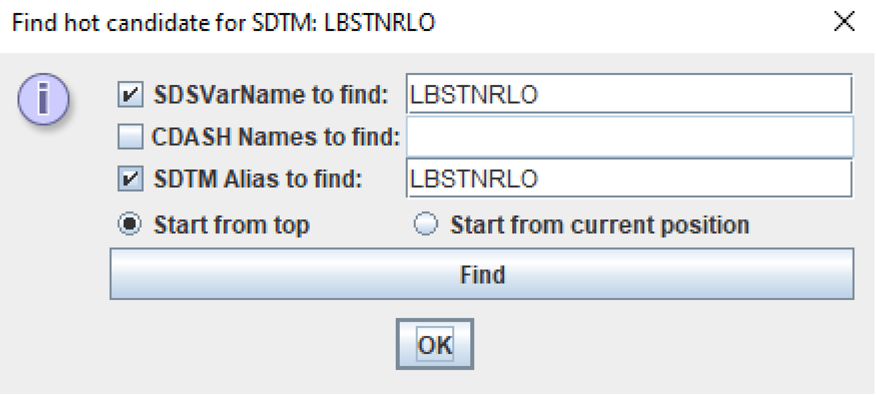
Alias: Synonyms in another context

Context	Synonym (name)
SDTM	LBORRES WHERE LBTESTCD=RBC

Also, when selecting an SDTM/SEND cell, all the "hot candidates" in the ODM tree are marked (with a rectangle on the "traffic light"), such as e.g. here for LBORNRLO (lower limit):


ItemGroupDef : Hematology
Description
ItemDef : Laboratory Name
ItemDef : Laboratory ID
ItemDef : Accession Number
ItemDef : RBC Done
ItemDef : RBC
ItemDef : RBC Normal Range Lo
ItemDef : RBC Normal Range Hi
ItemDef : WBC Done
ItemDef : WBC
ItemDef : WBC Normal Range Lo
ND
IT.SRNAM
RT
IT.LBORNRLO

And when using the menu "Navigate - Find hot SDTM candidate in ODM tree", one can even use a wizard allowing to further refine the search, e.g.:



Find hot candidate for SDTM: LBSTNRLO

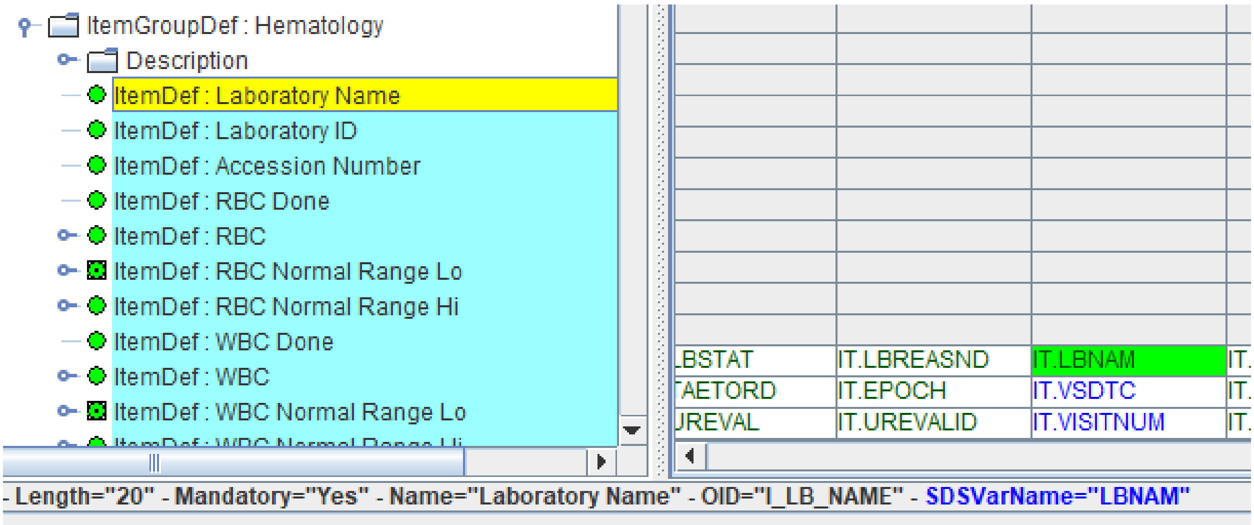
☑ SDSVarName to find: LBSTNRLO

☐ CDASH Names to find:

☑ SDTM Alias to find: LBSTNRLO

● Start from top ○ Start from current position

New is that also that “marking a hot candidate” can be done the other way around: when an item in the ODM tree is selected that is annotated with SDTM/SEND information, the corresponding cell in the SDTM/SEND table is highlighted (green when no mapping is already available, yellow when a mapping is already available), and the system automatically scrolls to the corresponding cell. For example, when clicking the item “Laboratory Name” in the ODM tree:



ItemGroupDef : Hematology
Description
ItemDef : Laboratory Name
ItemDef : Laboratory ID
ItemDef : Accession Number
ItemDef : RBC Done
ItemDef : RBC
ItemDef : RBC Normal Range Lo
ItemDef : RBC Normal Range Hi
ItemDef : WBC Done
ItemDef : WBC
ItemDef : WBC Normal Range Lo
ItemDef : WBC Normal Range Hi

LBSTAT	IT.LBREASND	IT.LBNAM	IT.
AETORD	IT.EPOCH	IT.VSDTC	IT.
JREVAL	IT.UREVALID	IT.VISITNUM	IT.

- Length="20" - Mandatory="Yes" - Name="Laboratory Name" - OID="I_LB_NAME" - SDSVarName="LBNAM"

the system highlights “LBNAM” cell on the right, and scrolls to it.
This new feature will further enormously help to exploit SDTM and also CDASH annotations for mapping with SDTM: the user then only needs to drag-and-drop and follow the wizards, which then automatically create the mapping script.

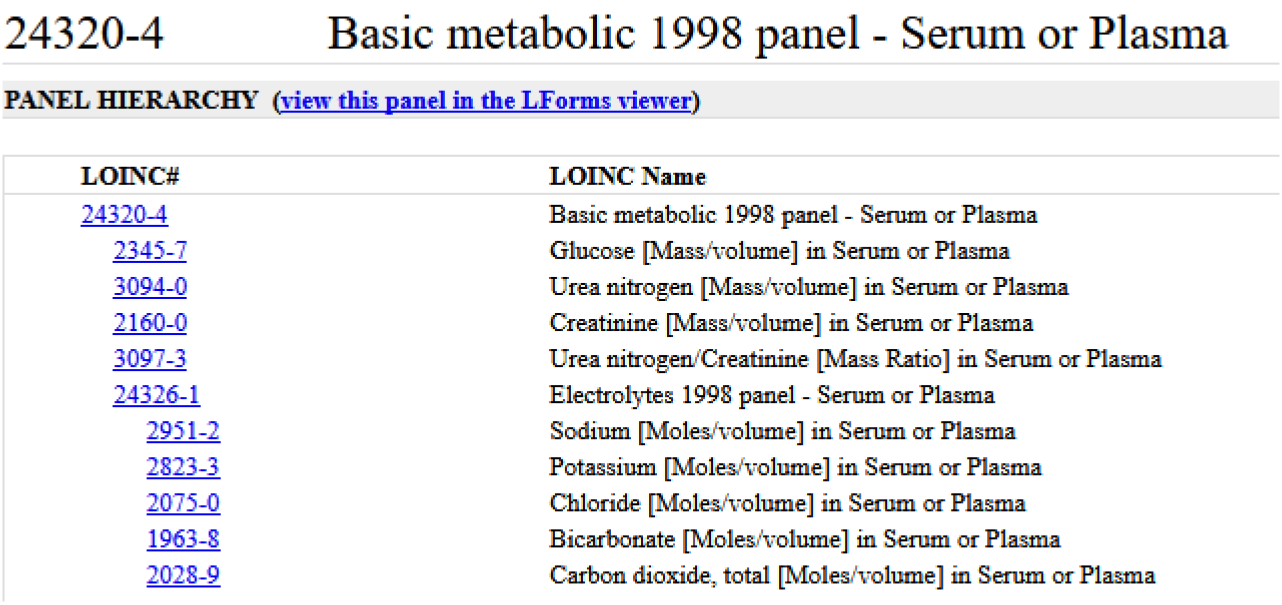
New functions in the mapping script language

Some new functions have been added to the scripting language (see document " Scripting_language_specification_v_4_0 .pdf"), including for using RESTful web services (see next section), and the separate tutorial "Working with RESTful Web Services".

Another useful function is the new "**alias()**" function, taking two arguments. The first argument is the OID of an Item, ItemGroup, Form or StudyEvent from the source data, the seconds the "Context" of the alias.

In ODM, the "Alias" element is used to define the synonym for an item in another context. Typically, this is used to assign the item a code, like a SNOMED-CT, RxNorm or LOINC code (the latter especially when the item represents a test).

For example, the protocol has stated that the "basic metabolic panel" lab tests must be executed. This consists of:

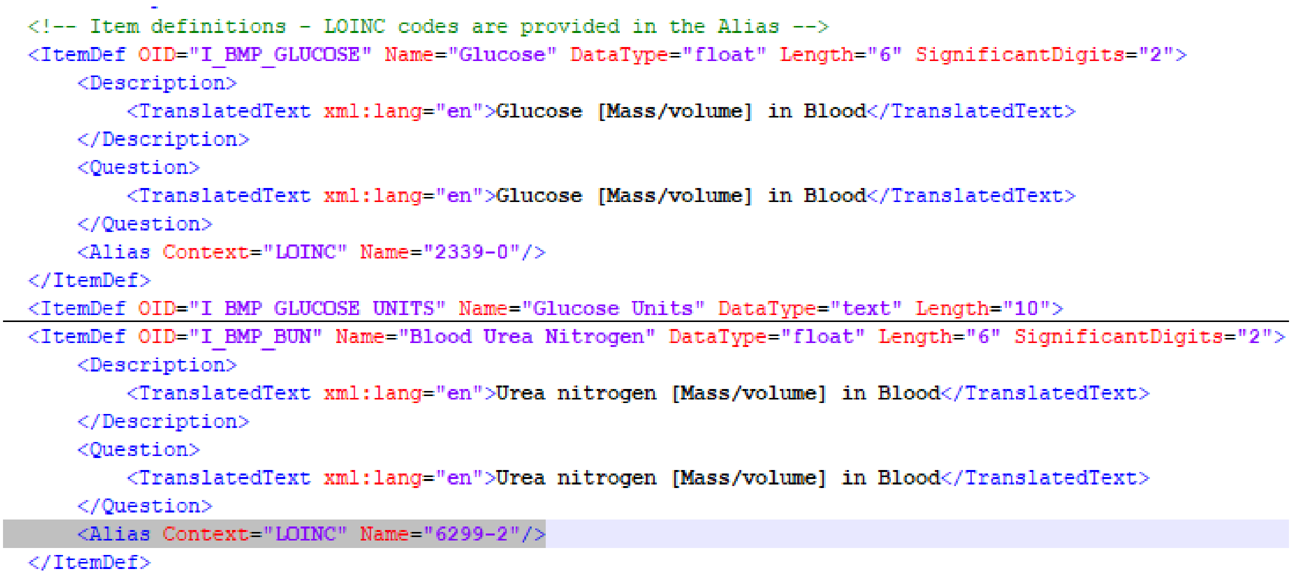


24320-4 Basic metabolic 1998 panel - Serum or Plasma

PANEL HIERARCHY (view this panel in the LForms viewer)

LOINC#	LOINC Name
24320-4	Basic metabolic 1998 panel - Serum or Plasma
2345-7	Glucose [Mass/volume] in Serum or Plasma
3094-0	Urea nitrogen [Mass/volume] in Serum or Plasma
2160-0	Creatinine [Mass/volume] in Serum or Plasma
3097-3	Urea nitrogen/Creatinine [Mass Ratio] in Serum or Plasma
24326-1	Electrolytes 1998 panel - Serum or Plasma
2951-2	Sodium [Moles/volume] in Serum or Plasma
2823-3	Potassium [Moles/volume] in Serum or Plasma
2075-0	Chloride [Moles/volume] in Serum or Plasma
1963-8	Bicarbonate [Moles/volume] in Serum or Plasma
2028-9	Carbon dioxide, total [Moles/volume] in Serum or Plasma

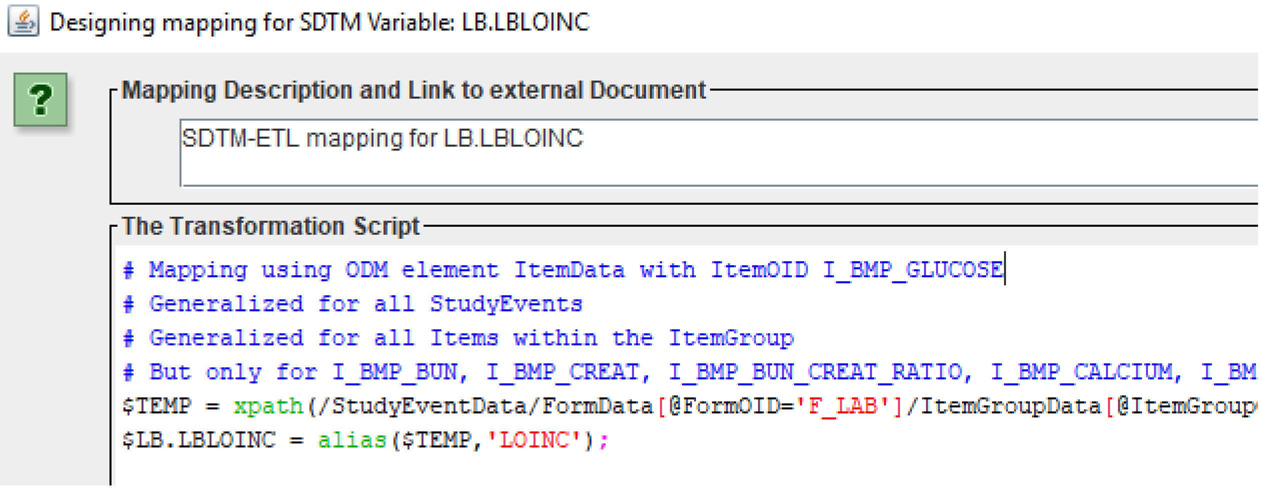
This is represented in the ODM study design e.g. as follows:



```
<!-- Item definitions - LOINC codes are provided in the Alias -->
<ItemDef OID="I_BMP_GLUCOSE" Name="Glucose" DataType="float" Length="6" SignificantDigits="2">
    <Description>
        <TranslatedText xml:lang="en">Glucose [Mass/volume] in Blood</TranslatedText>
    </Description>
    <Question>
        <TranslatedText xml:lang="en">Glucose [Mass/volume] in Blood</TranslatedText>
    </Question>
    <Alias Context="LOINC" Name="2339-0"/>
</ItemDef>
<ItemDef OID="I_BMP_GLUCOSE_UNITS" Name="Glucose Units" DataType="text" Length="10">
<ItemDef OID="I_BMP_BUN" Name="Blood Urea Nitrogen" DataType="float" Length="6" SignificantDigits="2">
    <Description>
        <TranslatedText xml:lang="en">Urea nitrogen [Mass/volume] in Blood</TranslatedText>
    </Description>
    <Question>
        <TranslatedText xml:lang="en">Urea nitrogen [Mass/volume] in Blood</TranslatedText>
    </Question>
    <Alias Context="LOINC" Name="6299-2"/>
</ItemDef>
```

i.e. each data point definition ("ItemDef") also contains an "Alias" element with "LOINC" as the "context", and the LOINC code as the name. This means that e.g. for "blood urea nitrogen", in the context of LOINC, the item is being defined as the test with code 6299-2.

The "alias()" function allows a lookup in the metadata for a data point, and to retrieve a code.
It has two arguments, the first being the ODM Item (usually using the XPath to it), the second being the context (i.e. the library of system, in the current case "LOINC").
This is especially interesting for e.g. populating "LBLOINC", as shown in the following mapping script:



Designing mapping for SDTM Variable: LB.LBLOINC

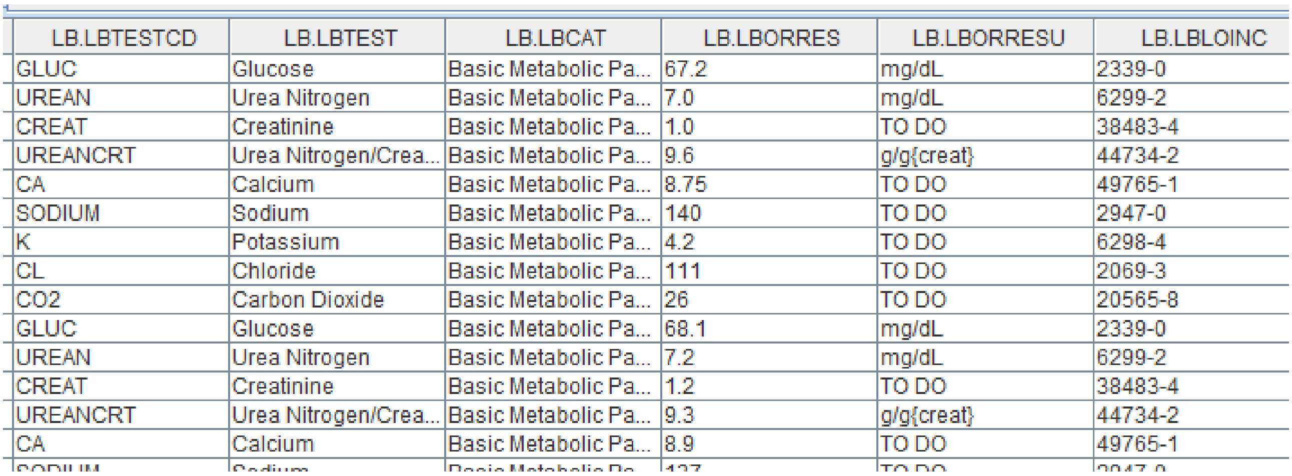
Mapping Description and Link to external Document

SDTM-ETL mapping for LB.LBLOINC

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_BMP_GLUCOSE
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# But only for I_BMP_BUN, I_BMP_CREAT, I_BMP_BUN_CREAT_RATIO, I_BMP_CALCIUM, I_BM
$TEMP = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroup
$LB.LBLOINC = alias($TEMP,'LOINC');
```

The line with "$TEMP = " picks up the OID of the test (an iteration over all lab tests is performed), and the "alias()" function then retrieves the value of the LOINC code from the metadata in the source ODM file.
The result is:



LB.LBTESTCD	LB.LBTEST	LB.LBCAT	LB.LBORRES	LB.LBORRESU	LB.LBLOINC
GLUC	Glucose	Basic Metabolic Pa...	67.2	mg/dL	2339-0
UREAN	Urea Nitrogen	Basic Metabolic Pa...	7.0	mg/dL	6299-2
CREAT	Creatinine	Basic Metabolic Pa...	1.0	TO DO	38483-4
UREANCRT	Urea Nitrogen/Crea...	Basic Metabolic Pa...	9.6	g/g{creat}	44734-2
CA	Calcium	Basic Metabolic Pa...	8.75	TO DO	49765-1
SODIUM	Sodium	Basic Metabolic Pa...	140	TO DO	2947-0
K	Potassium	Basic Metabolic Pa...	4.2	TO DO	6298-4
CL	Chloride	Basic Metabolic Pa...	111	TO DO	2069-3
CO2	Carbon Dioxide	Basic Metabolic Pa...	26	TO DO	20565-8
GLUC	Glucose	Basic Metabolic Pa...	68.1	mg/dL	2339-0
UREAN	Urea Nitrogen	Basic Metabolic Pa...	7.2	mg/dL	6299-2
CREAT	Creatinine	Basic Metabolic Pa...	1.2	TO DO	38483-4
UREANCRT	Urea Nitrogen/Crea...	Basic Metabolic Pa...	9.3	g/g{creat}	44734-2
CA	Calcium	Basic Metabolic Pa...	8.9	TO DO	49765-1
SODIUM	Sodium	Basic Metabolic Pa...	137	TO DO	2947-0

Working with RESTful Web Services

More and more, the use of RESTful web services for automating tasks in software is becoming custom, also in clinical research. Also CDISC is developing a number of RESTful web services for querying the CDISC Library metadata repository, and provides an RESTFUL-WS API for the CDISC Library.

How one can use the CDISC Library and its API is explained in a separate document "**Working with the CDISC Library in SDTM-ETL**".

In the field of RESTful web services for clinical research, XML4Pharma and the National Library of Medicine have been pioneers: both provide a number of free RESTful web services that can be used in software applications in clinical research.

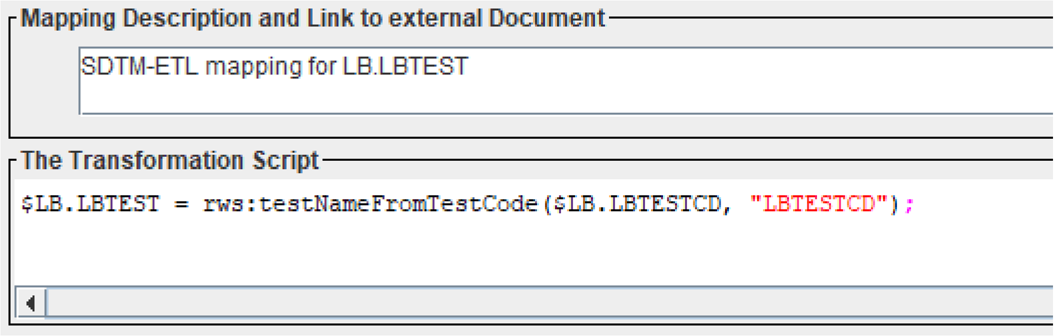
SDTM-ETL 4.0 comes with a number of pre-defined functions that use these RESTful web services, i.e. metadata information is requested from the XML4Pharma server about SDTM variables or LOINC codes.

For example, the function:

rws:testNameFromTestCode(String testCode, String variableName) can be used to obtain the "test name" (value for –TEST) for an SDTM variable that represents a test code (value for –TESTCD). For example, the function

rws:testNameFromTestCode("ALB", "LBTESTCD") will return "Albumin"

In a mapping script for LBTEST, this e.g. looks like:

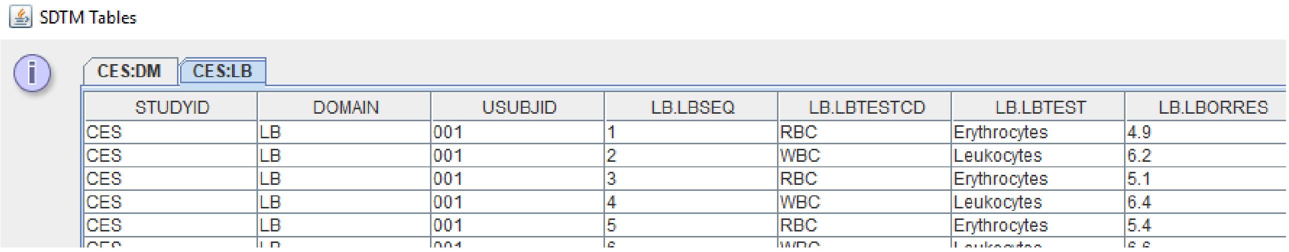

Mapping Description and Link to external Document

SDTM-ETL mapping for LB.LBTEST

The Transformation Script

```
$LB.LBTEST = rws:testNameFromTestCode($LB.LBTESTCD, "LBTESTCD");
```

And the result after execution for the LB dataset:



STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES
CES	LB	001	1	RBC	Erythrocytes	4.9
CES	LB	001	2	WBC	Leukocytes	6.2
CES	LB	001	3	RBC	Erythrocytes	5.1
CES	LB	001	4	WBC	Leukocytes	6.4
CES	LB	001	5	RBC	Erythrocytes	5.4

With "Erythrocytes" and "Leukocytes" being retrieved from "RBC" and "WBC" using the RESTful web service.

Another such pre-defined function is:

rws:sdtmLabel(String sdsVersion, String sdtmVariable)

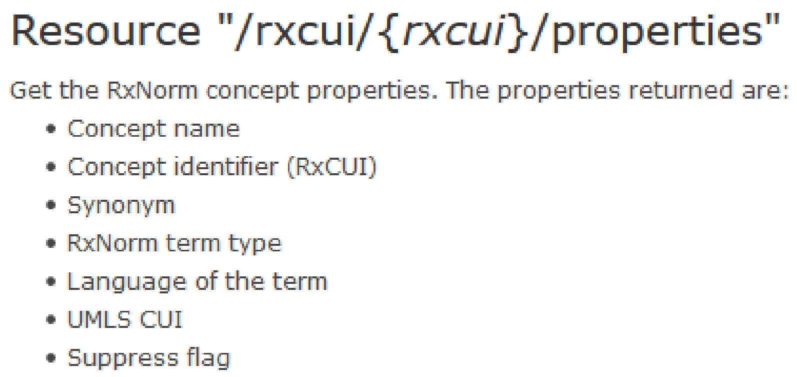
returning the "label" for the SDTM/SEND variable and SDS version (1.1, 1.2, 1.3)

For more details about these and all other predefined functions, please see the separate "**Using RESTful Web Services**" document.

One can however also use any other RESTful web services that are based on "HTTP GET", e.g. company-internal RESTful web services, or RESTful web services made available by the National Library of Medicine, e.g. for working with RxNorm medication numbers and codes.

For example, for getting the name of the medication with the RxNorm number "131725", one could use the NLM RESTful web service "properties" described by the NLM as:

RxNorm RESTful API



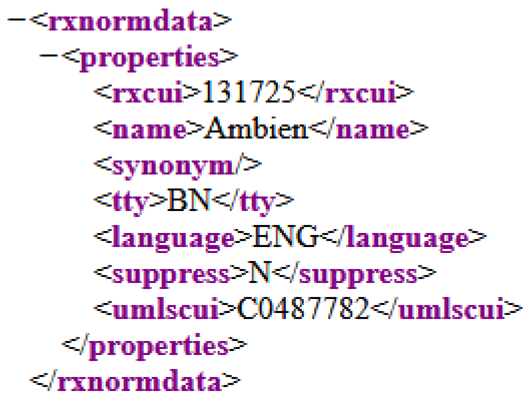
Resource "/rxcui/{*rxcui*}/properties"

Get the RxNorm concept properties. The properties returned are:

- Concept name
- Concept identifier (RxCUI)
- Synonym
- RxNorm term type
- Language of the term
- UMLS CUI
- Suppress flag

You can test this in the browser for RxNorm number 131725 by using:
https://rxnav.nlm.nih.gov/REST/rxcui/131725/properties

Delivering the xml:



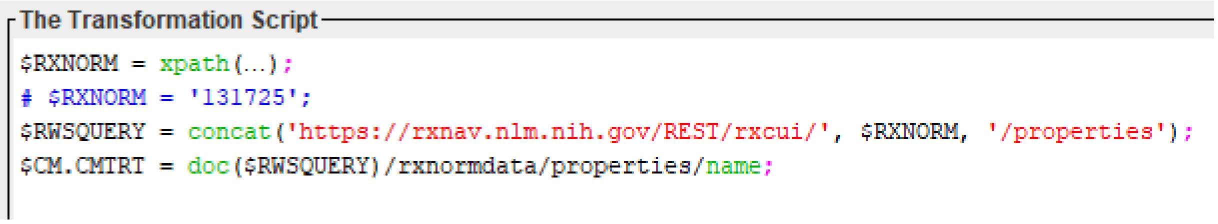
```
-<rxnormdata>
  -<properties>
      <rxcui>131725</rxcui>
      <name>Ambien</name>
      <synonym/>
      <tty>BN</tty>
      <language>ENG</language>
      <suppress>N</suppress>
      <umlscui>C0487782</umlscui>
    </properties>
  </rxnormdata>
```

However, you can also use this RESTful Web Service in your own mapping scripts, e.g. in "CMTRT" when the RxNorm medication number was e.g. retrieved from an electronic health record, or was collected as such on the CRF.

The mapping script would then be like:

$RXNORM = xpath(….);
$RWSQUERY = concat('https://rxnav.nlm.nih.gov/REST/rxcui/', $RXNORM, '/properties');
$CM.CMTRT = doc($RWSQUERY)/rxnormdata/properties/name;

In the mapping script editor:

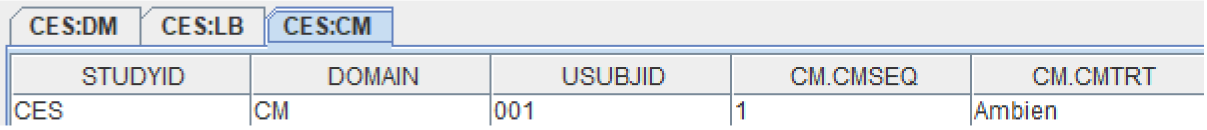


The Transformation Script

```
$RXNORM = xpath(...);
# $RXNORM = '131725';
$RWSQUERY = concat('https://rxnav.nlm.nih.gov/REST/rxcui/', $RXNORM, '/properties');
$CM.CMTRT = doc($RWSQUERY)/rxnormdata/properties/name;
```

Where the "doc" function means that an XML document is obtained when the $RWSQUERY is executed, and the "/rxnormdata/properties/name" the path (XPath) in the result document is.

This e.g. leads to the CM record for the case of RxNorm=131725:



STUDYID	DOMAIN	USUBJID	CM.CMSEQ	CM.CMTRT
CES	CM	001	1	Ambien

Limitation: at this moment, only RESTful web services that use HTTP or HTTPS and for which no authentication is necessary are supported.

Remark that it is always wise to store the "base" of the RESTful web service (in our case "'https://rxnav.nlm.nih.gov/REST/rxcui/'") in a "GLOBAL" variable for easy reuse.

For further details and possibilities, see the separate "**Using RESTful Web Services**" document.

Implementing the CDISC "LOINC-to-SDTM-LB" mappings

CDISC recently published the "final" version of a mapping between 2000+ LOINC codes (limited to the most popular test codes for laboratory tests, excluding microbiology tests) to SDTM-LB variables. The publication was in the form of an Excel file, which is of course not very useful for use in SDTM dataset generation, See the CDISC website for more details.

Therefore, we implemented the published mapping as a RESTful web service. We are currently also extending that mapping for a few thousand more laboratory LOINC codes, as out experience is that 2000 codes is a much too small sample (LOINC has over 93,000 codes).
Also, we developed mappings for the newly published COVID-19 test codes to the MB domain, and a set of LOINC codes for vital signs to the VS domain.

We then developed functions that use the RESTful web service for the LB domain use case.

The function that you will be using the most, is the function:

loinc2sdtmlb(string loinccode, string variablename)

The first parameter is the LOINC code, e.g. "1751-7" for "Albumin [Mass/Volume] in Serum or Plasma

The second parameter is the SDTM-LB variable you want the mapping value for.

For example, the by CDISC published mapping for LOINC code 1751-7 is:

SDTM variable	***SDTM variable value***
LBTESTCD	ALB
LBTEST	Albumin
LBSPEC	SERUM OR PLASMA
SUPPLB.LBPTFL	Y
SUP-PLB.LBRESTYP	MASS CONCENTRATION
SUPPLB.RSLSCL	QUANTITATIVE

So, the function call
loinc2sdtmlb('1751-7', 'LBTESTCD') will return 'ALB'
loinc2sdtmlb('1751-7', 'LBTEST') will return 'Albumin'
loinc2sdtmlb('1751-7', 'LBSPEC') will return 'SERUM OR PLASMA'
loinc2sdtmlb('1751-7', 'SUPPLB.LBRESTYP') will return 'MASS CONCENTRATION'

And so on. Variables for which no mapping was provided (e.g. for LBMETHOD in the case of LOINC code 1751-7) will return an empty value.

One can understand that this function, when correctly implemented, can enormously reduce the mapping effort for the LB domain and datasets, at least when the LOINC code of each of the tests is available.

An example of a set of mappings using this function (together with "Conventional" units to "SI" units conversion), where several of the variable values are automatically generated from the LOINC code can be found in the define.xml example file under

/TestFiles/define_2_0_mappings/ CES_LB_define_loinc2cdisc_testing.xml

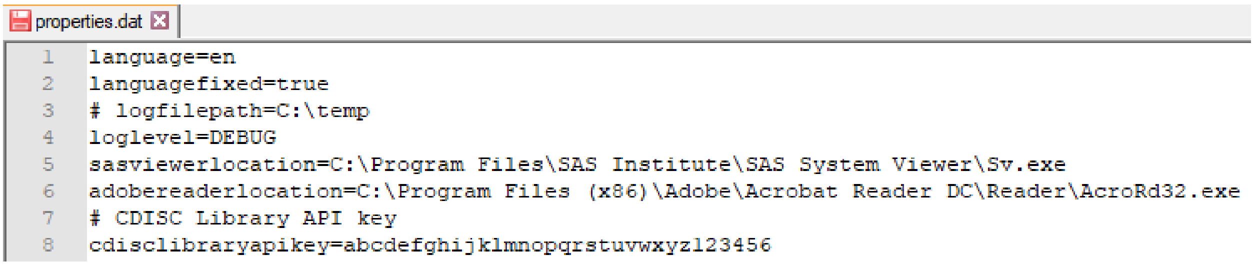
"US Conventional"- SI Unit conversions using the NLM RESTful web service

Performing unit conversions, especially for laboratory test results, has always been difficult, requiring a lot of programming. Such unit conversions are typically necessary for the generation of xxSTRESN values in SDTM and SEND. Using the RESTful web services developed by the National Library of Medicine (NLM), such conversions can be automated without programming. The new NLM RESTful web services do not only support classic conversions, such as from inches to centimeters, but also "US conventional units" to "SI units" conversions and vice versa, when either the LOINC code or the molecular weight is known. For the LB domain, the LOINC code of the test needs to be submitted anyway, so one has it already, making such automated conversions and population of the LBSTRESN variable very easy.

The functions for doing unit conversions using the NLM RESTful web service are further explained in the separate user manual "Performing Unit Conversions in SDTM-ETL".

CDISC Library

A number of the validation features (menu "Validate") use the CDISC Library API. The latter recently changed the authentication mechanism, from username and password, to "API key". For more details about the new authentication mechanism, please see the CDISC website.
As a consequence, we have adapted the software.
If you would like to use the "CDISC Library Validation" features, you will need such an API key, which is stored in the "properties.dat" file as a parameter value pair with the parameter name being "cdisclibraryapikey" (lower-case). For example:



```
language=en
languagefixed=true
# logfilepath=C:\temp
loglevel=DEBUG
sasviewerlocation=C:\Program Files\SAS Institute\SAS System Viewer\Sv.exe
adobereaderlocation=C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe
# CDISC Library API key
cdisclibraryapikey=abcdefghijklmnopqrstuvwxyz123456
```

Define.xml 2.0 to 2.1 transformation

On request of a number of customers, we added the feature enabling to fully automatically transform define.xml v.2.0 files to v.2.1. This can be done either from within SDTM-ETL, as in “stand-alone” mode.
This new feature is described in the separate manual “*Transforming define.xml 2.0 to define.xml 2.1*” which is also available on the SDTM-ETL website.

Bug fixes

- The element "BasicDefinitions" was, when present (e.g. using "Insert – MeasurementUnit definitions from ODM into define.xml") not automatically removed when "cleaning the define.xml" e.g. using "File – Save cleaned define.xml". This has been fixed.
- In case the location of the datasets is stored in the define.xml ("def:leaf") and the path to them is referencing a directory, and a "SASDatasetName" is stored, then the validation of the define.xml (using "Validate – define.xml) gave an error stating that the filename does not

correspond to the SAS dataset name. This has been fixed in the schematron.

Limitations

- Pinnacle21 does not support define.xml 2.1. As such, using Pinnacle21 in combination with define.xml 2.1 does not make sense. We hope that the CDISC CORE project delivers results early in 2022, so that the clinical research can finally switch to a much better validation software that does not produce the large numbers of false positives that Pinnacle21 produces.
  As soon as the open source CDISC CORE software becomes available, it will be implemented by us in SDTM-ETL.